НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему: Система прогнозування розвитку забруднень водяних(земних) поверхонь

Виконав: студент 4 курсу, групи ТВ-61

Трухан Денис Валерійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор, д.т.н. Бадаєв Ю.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент ст.в., к.т.н Воробйов М.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Трухану Денису Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи ”Система прогнозування розвитку забруднень водяних(земних) поверхонь”

керівник роботи професор, д.т.н Бадаєв Юрій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи персональний комп’ютер на операційній системі Windows 10, мова програмування Python, середовище розробки PyCharm.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати програмне забезпечення, аналогічне до розроблюваного, виявити його переваги та недоліки, розробити власний програмний продукт та користувацький інтерфейс.

!5. Перелік ілюстративного матеріалу користувацькі інтерфейси аналогічних програм, формули, приклади роботи програми, приклади застосування інструментів.

6. Публікації: _____

7. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.10.2019	
2.	Вивчення та аналіз задачі	11.10.2019- 23.12.2019	
3.	Розробка архітектури та загальної структури системи	03.02.2020- 04.03.2020	
4.	Розробка структур окремих підсистем	05.03.2020- 12.04.2020	
5.	Програмна реалізація системи	13.04.2020- 17.05.2020	
6.	Оформлення пояснювальної записки	08.05.2020- 07.06.2020	
7.	Захист програмного продукту	09.06.2020	
8.	Передзахист	09.06.2020	
9.	Захист	15.06.2020	

Студент _____ Трухан Д.В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Бадаєв Ю. І.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Робота містить 53 сторінок, 37 малюнків, 15 формул, 3 додатки та 15 посилань.

Мета роботи – створити програмне забезпечення для прогнозування розвитку забруднень водяних(земних) поверхонь.

У ході роботи було розглянуто PyCharm як середовище розробки. Використано мову програмування Python. Розглянуто аналогічну програму, виявлено її переваги та недоліки, розглянуто полікоординатні перетворення та бібліотеки Python. Розроблено програмний продукт для прогнозування розвитку забруднень водяних(земних) поверхонь.

Ключові слова: Python, PyCharm, полікоординатні перетворення, функції, графік.

ANNOTATION

The work contains 53 pages, 37 pictures, 3 appendices and 15 references.

The purpose of the work is to create software for forecasting the development of pollution of water (terrestrial) surfaces.

In the course of work PyCharm was considered as a development environment. Python programming language used. A similar program is considered, its advantages and disadvantages are revealed, polycoordinate transformations and Python libraries are considered. A software product for forecasting the development of pollution of water (land) surfaces has been developed.

Keywords: Python, PyCharm, polycoordinate transformations, functions, graph.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
Вступ 8
1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ ПРОГНОЗУВАННЯ 9
1.1 Призначення 9
1.2 Підсистеми 9
1.2.1 Функції 9
1.2.2 Користувацький інтерфейс 10
2. ОПИС ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ 10
2.1 Структура програми 10
2.2 Огляд існуючих систем 10
2.3 Огляд існуючих методів 12
2.4 Висновки до розділу 12
3. ЗАСОБИ РОЗРОБКИ 13
3.1 Мова програмування Python 13
3.2 Середовище розробки PyCharm 15
3.3 Технології, використані при розробці системи 17
3.4 Полікоординатні перетворення 23
3.4.1 Визначення полікоординатних перетворень 23
3.4.2 Варіанти політканинних перетворень 27
3.5 Користувацький інтерфейс 31
3.6 Висновки до розділу 35
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 36
4.1 Висновки 43
5. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ 44
5.1 Запуск системи 44
5.2 Інтерфейс користувача 44
5.3 Висновки 50
ВИСНОВКИ 51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Python – мова програмування

PyCharm – середовище розробки

AutoCad – система для проектування

Django - фреймворк

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

IDE – Integrated Development Environment

NumPy – Numerical Python

Tk – Tkinter

UI – User Interface

GUI – Graphical User Interface

LISP – LIst Processing Language

Вступ

Сучасна техногенна цивілізація, крім збільшення ступеня побутового комфорту, привела до стрімкого погіршення екологічної ситуації в світі. З часом зіпсована цивілізацією екологія може привести до катастрофічних наслідків.

Слово "Екологія" походить з грецької мови і означає "вивчення місця, де ми живемо". Отже, на Землі немає людини, яка б не впливала на екологічні проблеми. Наприклад, повітря, забруднене заводами пройде велику відстань разом з вітром, а важкі частинки потраплять у річки та моря.

Забруднення моря є найнебезпечнішим. Регіони, які використовуються для перевезень морськими шляхами мають найбільший показник забрудненості. Наприклад, Балтійське море – одна із найбрудніших зон на нашій планеті. Радіоактивний витік у Фукусімі, Японія, призвів до загибелі багатьох риб та інших водних істот.

Попередження екологічних надзвичайних ситуацій та катастроф - одна з найактуальніших проблем, яка стоїть перед вченими і інженерами. Екологічні надзвичайні ситуації на морі, як правило, супроводжуються забрудненням водного середовища нафтопродуктами. У цьому випадку завдання прогнозу розвитку надзвичайної ситуації зводяться до прогнозування поширення забруднення. В даний час існує досить велика кількість математичних моделей, що описують переміщення антропогенних домішок у водному середовищі. Застосовується комп’ютерні програми та системи для прогнозування розвитку забруднень. Одна із таких систем була реалізована у моєму дипломному проекті.

1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ ПРОГНОЗУВАННЯ

Сьогодні вчені намагаються створити програмне забезпечення для запобіганню розвитку забруднень нашої планети. Метою розробки мого дипломного проекту є реалізація системи розвитку забруднень земних та водяних поверхонь.

1.1 Призначення

Призначення даної системи є надання графічної інформації щодо забруднення земних та водяних поверхонь на основі введених даних користувачем. Система містить функції для вводу даних, які використовуються для побудови графіка і отримання кінцевого результату за допомогою спеціальних методів. Необхідними можливостями, які має забезпечувати система, є:

— зручний інтерфейс для введення даних користувачем;

— надання інформації у вигляді графіка з координатними осями;

— можливість збереження графіка у форматі PNG, PDF, SVG та інших;

— можливість задавати колір фігур;

— можливість задання розміру фігур;

— можливість задавати кількість кроків;

— можливість задавати вектори.

1.2 Підсистеми

1.2.1 Функції

Для обробки даних були використані функції, тобто для кожного блоку існує своя функція. Це робить код більш зручнішим для розуміння і читання.

1.2.2 Користувацький інтерфейс

Система була реалізована на мові програмування Python, тому для розробки інтерфейсу були використані спеціальні бібліотеки призначені для цього. Для вирішення задачі був зроблений вибір розробки програми, а не web-додатку. Для перезапуску програми не потрібно закривати її і заново відкривати. Інтерфейс має давати можливість користувачу вводити дані, перезапускати програму та виходити з неї.

2. ОПИС ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

2.1 Структура програми

Як було описано вище, програма написана за допомогою функцій, тому структура системи повністю відповідає функціям у коді. Функції зв’язані між собою так, як одна функція викликає інші функції. У головній функції, де визначені глобальні змінні, відбувається виклик функції, що відповідає за інтерфейс. З даної функції відбувається виклик функції, що відповідає за обробку даних, які вводить користувач. Також відбувається виклик функції, що відповідає за правильність введених даних у поля, іншими словами валідація полей.

Мабуть одна із важливіших функцій , це функція, де обробляються дані для отримання результату. Тобто отримавши дані користувача, функція обробляє їх, робить потрібні обчислення, після чого будує графік і користувач отримує результат.

2.2 Огляд існуючих систем

Данна система має аналог, який був розроблений на мові програмування LISP. Один із істотних недоліків даного аналога є те, що для запуску і використання

програми потрібна стороння платформа AutoCad будь-якої версії. AutoCAD використовується для створення автоматизованих конструкцій або програмних додатків. За допомогою AutoCad можна розробляти додатки у форматі 2D та 3D , надавати інструменти для проектування програмного забезпечення, яке використовується в архітектурі та управлінні проектами.

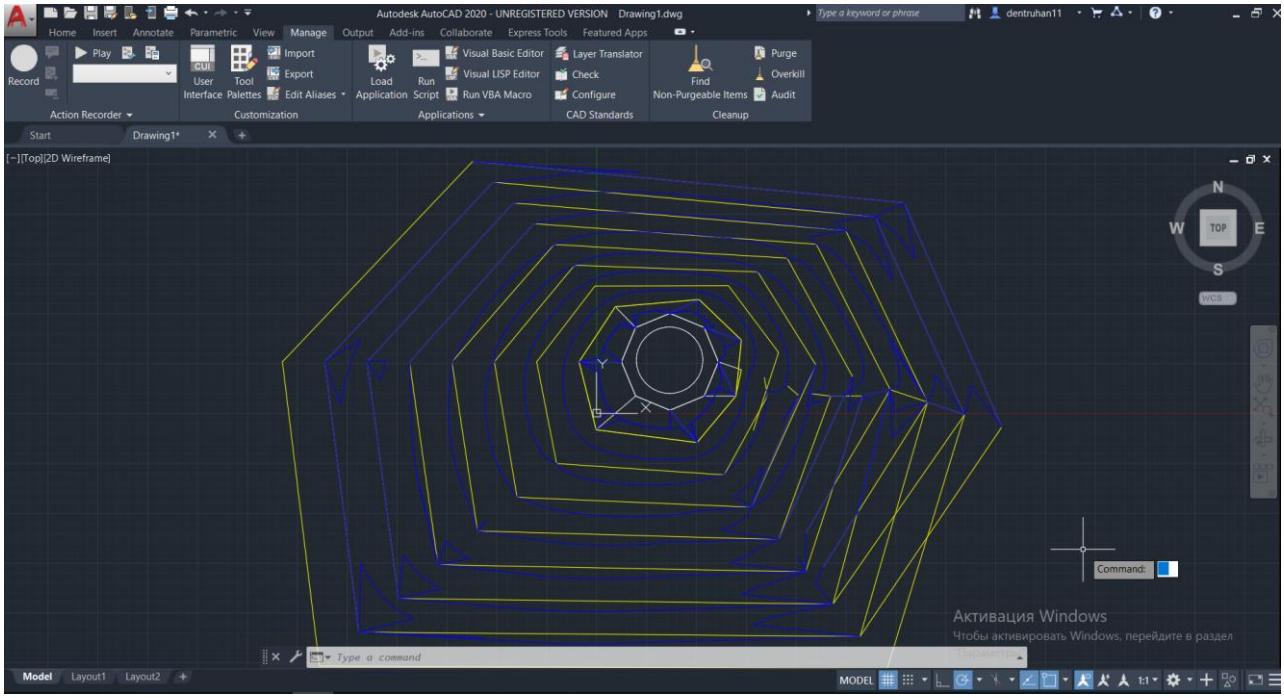

Рисунок 2.1 – Результат запуску аналогу

Для завантаження та для постійного використання AutoCad, потрібна платна підписка, що робить даний аналог незручним у використанні. Хоча використання AutoCad є і перевагою, тому що він має більш зручний та зрозумілий вивід результату, але потребується час для його завантаження. Моя система не потребується у завантаженні сторонніх платформ, тобто користувачу залишається всього лише натиснути на файл і програма запуститься. Також без спеціальної інструкції аналог не може бути правильно використан, тому що користувач повинен мати деякі навички для роботи з AutoCad .

2.3 Огляд існуючих методів

Відомі методи прогнозу розвитку забруднень в основному базуються на методах регресії, які застосовуються для кожної точки плями забруднення. Наприклад, один із таких методів – інформаційно-технологічний. Одною з переваг є відносно невелика кількість даних, головними із яких є регіональні гідрометеофактори.

Цей метод має такий недолік, що має достатньо громоздкий математичний аппарат, а також потребує великий обсяг робіт при програмній реалізації. Тому метою роботи було знайти такий метод, який мав би глобальний характер щодо плями забруднення. При розгляді різних методів з’ясувалося, що таким методом найбільш відповідає метод полікоординатних перетворень.

2.4 Висновки до розділу

У даному розділі було розглянуто аналог програми розробленої на мові програмування LISP. Встановлено переваги та недоліки один перед одним.

В якості існуючих програмних продуктів, під час роботи було розглянуто AutoCad. Наведено приклади інтерфейсу користувача в програмі. Виявлено переваги та недоліки цих програм. Описано чим розроблений продукт відрізняється від розглянутих.

Також було розглянуто існуючі методи для прогнозування розвитку забруднень. Було проведено аналіз, в ході якого з’ясувалось, що найбільш універсальним є метод полікоординатних перетворень.

3. ЗАСОБИ РОЗРОБКИ

Основним середовищем розробки було середовище PyCharm, яка надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. Також було використано середовище розробки IDLE, яка була створена за допомогою бібліотеки Tk. Ця бібліотека надає змогу використовувати IDLE на багатьох операційних системах, наприклад, Windows, Mac OS та Unix-подібних ОС.

Для розробки безпосередньо програми використовувалась мова програмуваня Python версією 3.8.2. Python - це високорівнева мова програмування загального призначення, яка використовується в тому числі і для розробки веб-додатків. Мова орієнтована на підвищення продуктивності розробника і читання коду. Python підтримує кілька парадигм програмування: структурний, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. У мові є динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Програмний код на Python організовується у функції та класи, які можуть об'єднуватися в модулі, а вони в свою чергу можуть бути об'єднані в пакети.

3.1 Мова програмування Python

Python - популярна мова програмування, створена Гідо ван Россумом у 1991 році, яка використовується у багатьох сферах:

— веб-розробка (на стороні сервера);

— розробка програмного забезпечення;

— математика;

— системний сценарій.

Також Python має такі можливості:

— створення веб-додатків на сервері;

— створення робочих процесів;

— підключення до систем баз даних.

— можна читати та змінювати файли.

— обробка великих даних та виконання складної математики.

— Швидка розробка програмного забезпечення, готового до виробництва.

Python працює на різних платформах (Windows, Mac, Linux, Raspberry Pi тощо). У Python є простий синтаксис, подібний до англійської мови. У Python є синтаксис, який дозволяє розробникам писати програми з меншою кількістю рядків, ніж деякі інші мови програмування.

Python працює в системі інтерпретатора, тобто код може бути виконаний, як тільки він буде записаний. Це означає, що прототипування може бути дуже швидким.

Найновішою основною версією Python є Python 3, яка використовується при написанні програми. Однак Python 2, хоча не оновлюється нічим іншим, окрім оновлень безпеки, все ще залишається досить популярним.

Python може буде записаний у текстовому редакторі. Можна писати Python в інтегрованому середовищі розробки, таких як Thonny, Pycharm, Netbeans або Eclipse, які особливо корисні при керуванні більшими колекціями файлів Python.

Синтаксис Python трохи схожий на інші мови програмування, але все одно має свої переваги порівняно з іншими мовами:

— Python був розроблений для читабельності і має деякі схожість з англійською мовою з впливом математики;

— Python використовує нові рядки для завершення команди, на відміну від інших мов програмування, які часто використовують крапки з комою або круглими дужками;

— для визначення області застосування Python покладається на відступ, використовуючи пробіли, наприклад, область циклів, функції та класи. Інші мови програмування часто використовують для цієї мети фігурні дужки.

Python порівняно простий, тому його легко вивчити, оскільки він вимагає унікального синтаксису, який зосереджений на читанні. Розробники можуть читати та перекладати код Python набагато простіше, ніж інші мови. У свою чергу, це зменшує витрати на підтримку та розробку програми, оскільки дозволяє командам працювати спільно без значних бар'єрів у мові та досвіді.

Крім того, Python підтримує використання модулів і пакетів, а це означає, що програми можуть бути спроектовані в модульному стилі, а код можна використовувати повторно в різних проектах. Розробивши потрібний вам модуль або пакет, його можна масштабувати для використання в інших проектах, і легко імпортувати або експортувати ці модулі.

Однією з найбільш перспективних переваг Python є те, що і стандартна бібліотека, і інтерпретатор доступні безкоштовно, як у двійковій, так і у вихідній формі. Ексклюзивності також немає, оскільки Python та всі необхідні інструменти доступні на всіх основних платформах. Тому це привабливий варіант для розробників, які не хочуть турбуватися про оплату високих витрат на розробку.

3.2 Середовище розробки PyCharm

PyCharm - це гібридна платформа, розроблена JetBrains як IDE для Python, що забезпечує широкий спектр необхідних інструментів для розробників Python, щоб створити зручне середовище для продуктивної розробки програм на мові програмування Python. Він підтримує дві версії: v2.x та v3.x.

Ми можемо запускати PyCharm в Windows, Linux або Mac OS. Крім того, він містить модулі та пакети, які допомагають програмістам розробити програмне забезпечення, використовуючи Python, за менший час та з мінімальними зусиллями.
Деякі основні функції, надані PyCharm:

1. Розумний редактор коду:

— завдяки цьому програмування стає більш зручнішим та швидшим;

— складається з кольорових виділень для ключових слів, класів та функцій;

— допомагає збільшити читабельність та розуміння коду;

— допомагає легко визначити помилки;
— надає функцію автозаповнення та інструкції щодо заповнення коду.

2. Навігація по коду:
— допомагає розробникам в редагуванні та вдосконаленні коду з меншими зусиллями та часом;
— за допомогою навігації з кодом розробник може легко перейти до функції, класу чи файлу;
— програміст може знайти елемент, символ або змінну у вихідному коді протягом найкоротшого часу;
— крім того, використовуючи режим об'єктива, розробник може ретельно перевірити і налагодити весь вихідний код.

3. Рефакторинг:
— має перевагу в тому, щоб зробити ефективні та швидкі зміни як локальних, так і глобальних змінних;
— рефакторинг в PyCharm дозволяє розробникам вдосконалити внутрішню структуру, не змінюючи зовнішню продуктивність коду;
— також допомагає розділити більш розширені класи та функції за допомогою методу вилучення.

4. Допомога для багатьох інших веб-технологій:
— допомагає розробникам створювати веб-додатки в Python;
— підтримує популярні веб-технології, такі як HTML, CSS та JavaScript;
— розробники мають вибір редагування в реальному часі за допомогою цього IDE.
— Одночасно вони можуть переглядати створену / оновлену веб-сторінку;
— розробники можуть стежити за змінами безпосередньо у веб-браузері;
— PyCharm також підтримує AnglularJS та NodeJS для розробки веб-додатків.

5. Підтримка популярних фреймворків Python:

— PyCharm підтримує фреймворки, такі як Django;
— надає функцію автозаповнення щодо параметрів Django;
— допомагає в налагодженні кодів Django;
— також підтримує web2py та Pyramid, інші популярні фреймворки.

6. Підтримка бібліотек Python:
— PyCharm підтримує бібліотеки Python, такі як Matplotlib, NumPy та Anaconda;
— бібліотеки допомагають будувати проекти Data Science та Machine Learning;
— складається з інтерактивних графіків, які допомагають розробникам розуміти дані;
— здатний інтегруватися з різними інструментами, такими як IPython, Django та Pytest. Ця інтеграція допомагає впроваджувати унікальні рішення.
Під час розробки системи була використана версія PyCharm 2019.3.3.

3.3 Технології, використані при розробці системи

Для розробки системи було використано декілька бібліотек Python, а саме Matplotlib, Matplotlib.pyplot, Matplotlib.lines, Matplotlib.path – для візуалізації результату та Math – для обчислення формул.

Matplotlib – це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python, яка була розроблена Джоном Хантером, який разом із численними учасниками вкладав безмежну кількість часу та зусиль, щоб тисячі вчених та звичайних людей використовували її у своїх цілях.

Matplotlib створює фігури у різних форматах та інтерактивних середовищах на різних платформах. Matplotlib може використовуватися в скриптах Python, оболонці Python та IPython, серверах веб-додатків та різних інструментах графічного інтерфейсу користувача.

Matplotlib має багато переваг над іншими схожими бібліотеками для створення візуалізацій. Ось деякі функції та переваги даної бібліотеки:

— можливість створювати діаграми, графіки, фігури за допомогою лише кількох рядків коду;

— можливість використовувати інтерактивні фігури, які можна збільшувати, зменшувати та змінювати ;

— повний контроль над стилями рядків, властивостями шрифту, властивостями осей;

— можливість експортувати та зберігати в деяких форматах файлів ;

— можливість дослідити спеціалізовані функції, що надаються сторонніми пакетами;

— можливість дізнатися більше про Matplotlib за допомогою багатьох зовнішніх навчальних ресурсів;

— можливість створювати 3D-фігури.

Matplotlib поставляється з декількома додатковими наборами інструментів, включаючи 3D-графіки з mplot3d, помічниками осей у axes_grid1 та помічниками осей у axartist.

Велика кількість сторонніх пакетів розширюється та ґрунтується на функціональності Matplotlib, включаючи кілька графічних інтерфейсів вищого рівня (seaborn, HoloViews, ggplot, ...) та два набори інструментів для проектування та картографування (Basemap та Cartopy).

Пакет підтримує багато видів графіків і діаграм:

— графіки (line plot)

— діаграми розкиду (scatter plot);

— стовпчасті діаграми (bar chart) і гістограми (histogram);

— кругові діаграми (pie chart);

— стовбур-лист діаграми (stem plot);

— контурні графіки (contour plot);

— поля градієнтів (quiver);

— спектральні діаграми (spectrogram).

Matplotlib складається з безлічі модулів. Модулі наповнені різними класами і функціями, які ієрархічно пов'язані між собою. Однією з ключових особливостей matplotlib, яку я хотів би підкреслити, і що я вважаю робить matplotlib дуже зручним для створення фігур для наукових публікацій, - це те, що всі аспекти фігури можна

керувати програмно. Це важливо для відтворюваності та зручно, коли потрібно змінити фігуру за допомогою оновлених даних або змінити її зовнішній вигляд.

Також у дипломному проєкті було використано один із модулів бііліотеки Matplolib, назва якого Pyplot. Pyplot - це модуль Matplotlib, який забезпечує MATLAB-подібний інтерфейс. Matplotlib розроблений так само, як MATLAB, з можливістю використання Python та перевагою бути вільним та відкритим. Кожна функція pyplot вносить певні зміни до фігури: наприклад, створює фігуру, створює графічну площу на фігурі, накреслює деякі рядки в зоні побудови графіків, прикрашає ділянку мітками тощо. За допомогою pyplot можна створити графік, діаграму, гістограму, 3D-графік, зображення, контур та полярність. Pyplot робить розробку системи більш зручнішою та потребує малу кількість рядків для побудови графіків:

```
1  import matplotlib.pyplot as plt
2
3  plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
4  plt.axis([0, 6, 0, 20])
5  plt.show()
6
```

Рисунок 3.1 – Код для побудови графіка

Результат програми:

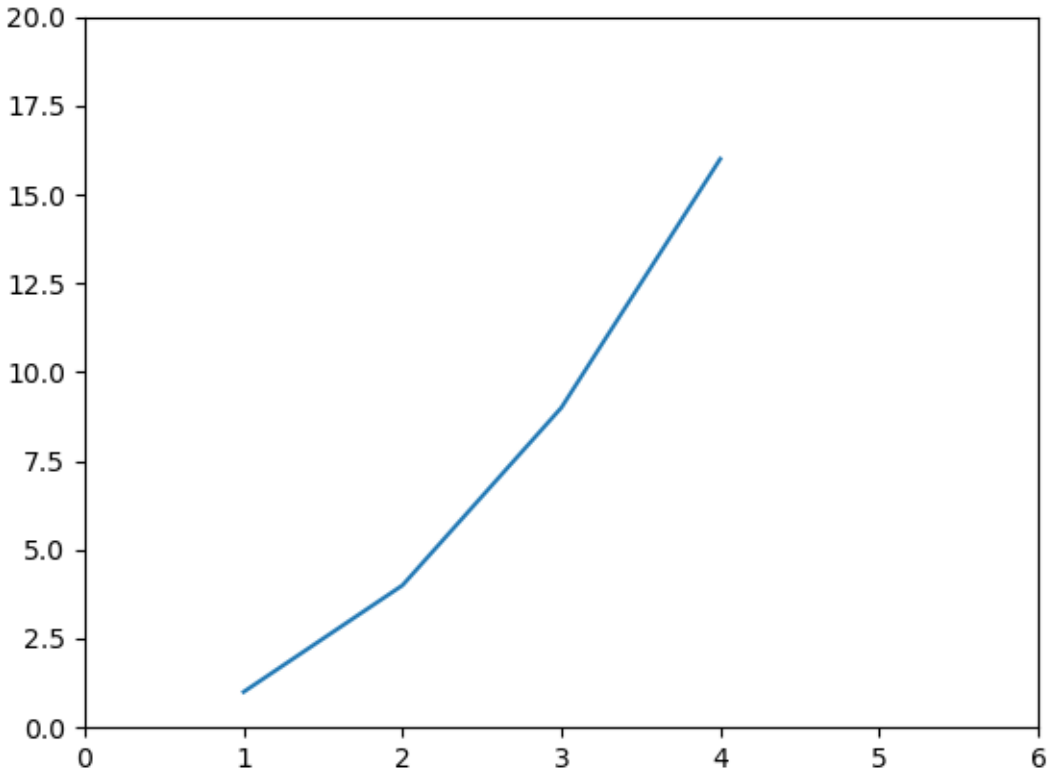

Рисунок 3.2 – Результат програми

Функція графіку позначає x-координати (1, 2, 3, 4) та y-координати (1, 4, 9, 16) у лінійному графіку із заданими масштабами. Ця функція приймає параметри, які дозволяють нам встановлювати масштаби осей і форматувати графіки. Ці параметри згадані нижче:

— plot(x, y): графік x і y, використовуючи стиль та колір лінії за замовчуванням;

— plot.axis([xmin, xmax, ymin, ymax]): масштабує вісь x і вісь y від мінімальних до максимальних значень;

— plot.(x, y, color=’green’, marker=’o’, linestyle=’dashed’, linewidth=2, markersize=12): координати x і y позначаються круговими маркерами розміром 12 і зеленою лінією;

— plot.xlabel(‘X-axis’): називає вісь x;

— plot.ylabel(‘Y-axis’): називає вісь y.

Для наступного прикладу ми будемо використовувати набори даних про споживання електроенергії Індії та Бангладеш. Тут ми використовуємо Загальнодоступні дані Google як джерело даних. Код прикладу:

```
import matplotlib.pyplot as plt

year = [1972, 1982, 1992, 2002, 2012]
e_india = [100.6, 158.61, 305.54, 394.96, 724.79]
e_bangladesh = [10.5, 25.21, 58.65, 119.27, 274.87]

plt.plot(year, e_india, color='orange',
         label='India')
plt.plot(year, e_bangladesh, color='g',
         label='Bangladesh')

plt.xlabel('Years')
plt.ylabel('Power consumption in kWh')

plt.title('Electricity consumption per capita\
of India and Bangladesh')
plt.legend()
plt.show()
```

Рисунок 3.3 – Код для побудови графіку споживання електроенергії

Результат програми:

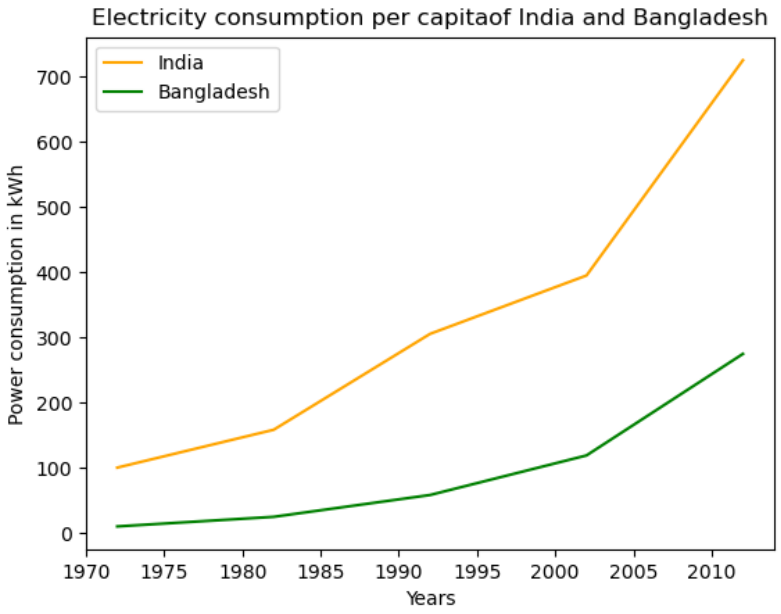

Рисунок 3.4 – Результат виконаного коду

Отже, створення малюнка в matplolib є дуже простим і зручним. Написання коду прикладів, знаючи методи даної бібліотеки, простіше, ніж зробити це за допомогою сторонніх сервісів.

Також для обчислення геометричних формул було використано бібліотеку Math. Математичний модуль Python - важлива особливість, розроблена для роботи з математичними операціями. Він поставляється в комплекті зі стандартним випуском Python. Більшість функцій математичного модуля - це тонкі обгортки навколо математичних функцій платформи C. Оскільки основні його функції записані в CPython, математичний модуль ефективний і відповідає стандарту C. Модуль математики Python дає можливість виконувати загальні та корисні математичні обчислення у вашій програмі. Ось кілька практичних застосувань для математичного модуля:

— обчислення комбінацій та перестановок за допомогою факторіалів;

— обчислення висоти полюса за допомогою тригонометричних функцій;

— обчислення радіоактивного розпаду за допомогою експоненціальної функції;

— обчислення кривої підвісного моста за допомогою гіперболічних функцій;

— розв’язування квадратичних рівнянь;

— моделювання періодичних функцій, таких як звукові та світлові хвилі, використовуючи тригонометричні функції.

Для обчислення системи рівнянь був використан пакет NumPy. NumPy - бібліотека, яка використовується для роботи з масивами. Вона також має функції для роботи в області лінійної алгебри, перетворення Фур'є та матриць. NumPy був створений у 2005 році Тревісом Оліфантом. Це проект з відкритим кодом, і ним можна вільно користуватися. NumPy означає Numerical Python. Навіщо використовувати NumPy? У Python у нас є списки, які служать цілі масивів, але вони обробляються повільно. NumPy спрямований на надання об’єкта масиву, який на 50 разів швидше, ніж перераховані традиційні Python. Об'єкт масиву в NumPy називається ndarray, він надає безліч підтримуючих функцій, що робить роботу з ndarray дуже простою. Масиви дуже часто використовуються в науці даних, де

швидкість і ресурси дуже важливі. Data Science: це галузь інформатики, де ми вивчаємо, як зберігати, використовувати та аналізувати дані для отримання інформації з неї. Чому NumPy швидше списків? Масиви NumPy зберігаються на одному постійному місці в пам'яті на відміну від списків, тому процеси можуть отримувати доступ до них та керувати ними дуже ефективно. Така поведінка в інформатиці називається місцем відліку. Це головна причина, чому NumPy швидше, ніж списки. Також він оптимізований для роботи з найновішими архітектурами процесора. NumPy є бібліотекою Python і частково пишеться на Python, але більшість частин, які потребують швидких обчислень, написані на C або C ++.

В роботі наведено метод полікоординатних перетворень точок, який заключається в наступному.

3.4 Полікоординатні перетворення

3.4.1 Визначення полікоординатних перетворень

Визначення. Лінійною політканиною розміру ***p*** у просторі $\boldsymbol{R^2}$ будемо називати сукупність ***p*** *сімей* прямих виду (див. рис. 3.5-а при p=5):

$$\beta_i = a_i x + b_i y + c_i, i = 1,2,..., p \geq 3. \qquad (3.1)$$

Джерело [12,13]

Нехай на площині в декартовій системі координат xOy задана політканина (***p***-*тканина*) за допомогою ***p*** лінійних функцій-координат (3.1). Перетворимоцю ***p***-*тканину*, тобто змінемо деяким чином положення координатних функцій. Нова ***p***-*тканина* буде визначена новою системою рівнянь (3.1):

$$\varphi_i = A_i x' + B_i y' + C_i, i = 1,2,..., p \geq 3 \ . \qquad (3.2)$$

Джерело [12,13]

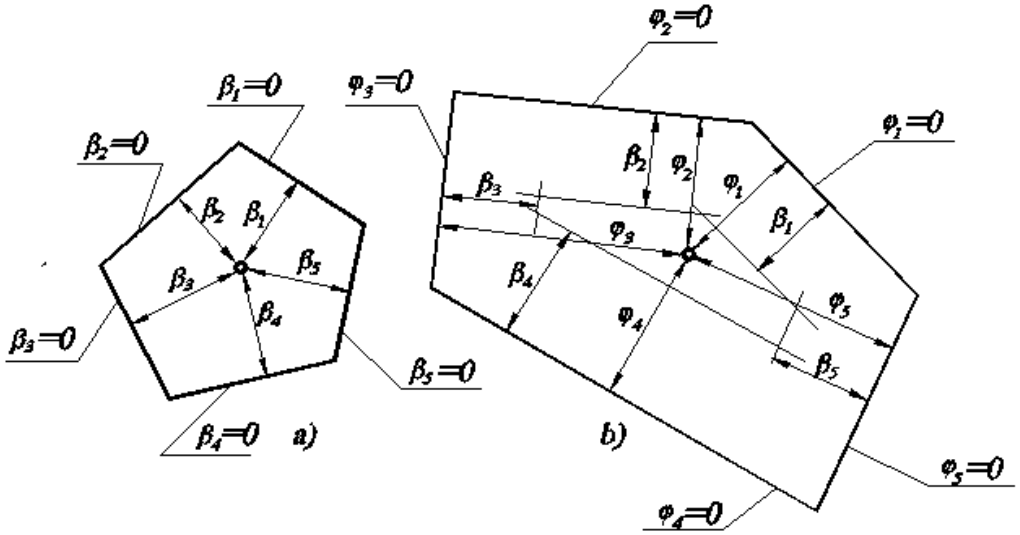

Рис. 3.5. Політканинні перетворення точки

Джерело [12,13]

Положення будь-якої точки плошини в попередній ***p-тканині визначається*** координатами (x,y) — в системі координат xOy — або також сукупністю координат β_i, i=1,2,...,p≥3. Аналогічно, після перетворення політканини — (x',y') и φ_i, i=1,2,...,p≥3. Якщо розглядати конкретну точку, то її політканинні координати до перетворення і після не співпадуть тобто $\beta_i \neq \varphi_i$, i=1,2,...,p≥3. Якщо ж прийняти протилежне, тобто намагатись побудувати точку в перетвореній *p*-тканині при умові$\beta_i=\varphi_i$, i=1,2,...,p≥3, то замість точки отримаємо багатокутник, конфигурація якого одднозначно визначається орієнтацією координатних функцій політканини (3.2). Таким чином, виникає необхідність в тому, щоб отримати однозначний розв'язок задачі політканинного перетворення площини, тобто встановлення функціональної взаємозв'язку між координатами β_i и φ_i, i=1,2,...,p≥3 обох політканин.

Приймаючи до уваги те, що координата точки в політканині є аналогом її віддаленості від відповідної координатної лінії, можна записати:

$$\varphi_i = \omega_i \beta_i, i = 1,2,\ldots, p \geq 3 \quad . \tag{3.3}$$

Джерело [12,13]

Таким чином, (3.2) буде виглядати:

$$\omega_i \beta_i = A_i x + B_i y + C_i, i = 1,2,..., p \geq 3 \quad . \tag{3.4}$$

Джерело [12,13]

Система (3.4) має ***p*** рівнянь і ***p+2*** невідомих (ω_i, i=1,2,...p і x,y). Таким чином, для того, щоб отримати однозначний розв'язок(6.4), необхідно ввести дві додаткові умови. Пропонується знайти ω_i, i=1,2,...p за умовою їх мінімального відхилення від ω_j j≠i, наприклад, реалізувати наступний функціонал :

$$S = \sum_{i=1}^{p} \sum_{j=1}^{p} (\omega_i - \omega_j)^2 \longrightarrow \min, p \geq 3 \tag{3.5}$$

Джерело [12,13]

За допомогою диференціювання функціоналу (3.5) знайдемо його экстремум. Для цього необхідно встановити змінні, за якими буде виконуватись диференціювання.

Перепишемо (3.4) в наступному вигляді:

$$\omega_i = (A_i x' + B_i y' + C_i) / \beta_i, i = 1,2,..., p \geq 3. \tag{3.6}$$

Джерело [12,13]

Таким чином, $\omega_i=f(x',y')$, i=1,2,...p≥3 , тобто змінними диференціювання функціоналу (3.5) будуть x' і y'.

Знайдемо умови досягнення экстремума функціонала:

$$\begin{cases} \dfrac{\partial S}{\partial x'} = 2\sum\limits_{i=1}^{p}\ \sum\limits_{j=1}^{p}(\omega_i-\omega_j)\left(\dfrac{\partial \omega_i}{\partial x'}-\dfrac{\partial \omega_j}{\partial x'}\right)=0, \\ \dfrac{\partial S}{\partial y'} = 2\sum\limits_{i=1}^{p}\ \sum\limits_{j=1}^{p}(\omega_i-\omega_j)\left(\dfrac{\partial \omega_i}{\partial y'}-\dfrac{\partial \omega_j}{\partial y'}\right)=0. \end{cases} \qquad (3.7)$$

Джерело [12,13]

Підставивши в (3.7) ω_i, i=1,2,...,p≥3 із (6.6), отримаємо дві додаткові умови для однозначного розв'язку системи (6.4) в наступному вигляді:

$$\begin{cases} D_x x' + D_y y' + D = 0, \\ G_x x' + G_y y' + G = 0, \end{cases} \qquad (3.8)$$

Джерело [12,13]

де:

$$D_x = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{A_i}{\beta_i}-\frac{A_j}{\beta_j}\right)^2, \quad G_x = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{A_i}{\beta_i}-\frac{A_j}{\beta_j}\right)\left(\frac{B_i}{\beta_i}-\frac{B_j}{\beta_j}\right),$$

$$D_y = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{A_i}{\beta_i}-\frac{A_j}{\beta_j}\right)\left(\frac{B_i}{\beta_i}-\frac{B_j}{\beta_j}\right), \quad G_y = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{B_i}{\beta_i}-\frac{B_j}{\beta_j}\right)^2,$$

$$D = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{A_i}{\beta_i}-\frac{A_j}{\beta_j}\right)\left(\frac{C_i}{\beta_i}-\frac{C_j}{\beta_j}\right), \quad G = \sum_{i=1}^{p}\ \sum_{j=1}^{p}\left(\frac{B_i}{\beta_i}-\frac{B_j}{\beta_j}\right)\left(\frac{C_i}{\beta_i}-\frac{C_j}{\beta_j}\right).$$

i=1,2,...,p≥3;

Джерело [12,13]

Систем (3.8) дозволяє отримати розв’язок у вигляді координат точки (x’,y’) після перетворення політканини. У випадку необхідності, змінні ω_i i=1,2,...p≥3 можуть бути визначені за знайденими координатами (x’,y’) за допомогою систем (3.6) и (3.3) відповідно.

Таким чином, політканинні перетворення визначається на основі мінімізації функціонала (3.5). Як бачимо, що таке перетворення встановлює відповідність двох точкових полів. Можно бачити, що проективна геометрія є частковим випадком політканинних перетворень, а саме при p=3 і $\omega_1=\omega_2=\omega_3$. В деякому сенсі можна говорити о політканинних геометріях, які визначаються полікоординатними перетвореннями. Важкість визначається у встановленні геометричних інваріантів. В якості інваріанта тут може виступати тільки функціонал (3.5), який встановлює відповідність точкових полів. Можна сказати про те, що із збільшенням параметрів лінійних геометрій (евклідова, афінна, проективна) кількість геометричних інваріантів зменшується і в проективній геометрії це тільки мало зрозуміле з геометричної точки зору складне співвідношення чотирьох точок.

3.4.2 Варіанти політканинних перетворень

До сих пір був показаний тільки один із варіантів політканинних перетворень. Можна варіювати функціонал (3.5), що дасть різноманітні варіанти політканинних перетворень. Далі від показаного варіанта плоскої політканини легко перейти к тривимірному і багатовимірного простру. Крім того, можна застосувати вагові коефіцієнти, різні для кожної політканинної координати, що дасть специфічні особливості перетворень. Крім того, від лінійних політканинних перетворень можна перейти до нелінійних, хоча такий варіант сильно ускладнить розв’язок.

Розглянемо різні варіанти функціонала.

Функціонал (3.5) достатньо трудомісткий для обчислень. Спростимо його наступним чином:

$$S = \sum_{\text{i}=1}^{p} (\omega_{\text{i}} - M)^2 \longrightarrow \min,\ p \geq 3, \tag{3.9}$$

Джерело [12,13]

де М – аналог масштабного множника. Величини ω_{i} будуть наближатись до цього множника.

Продиференціюємо (3.9). Отримаємо:

$$\begin{cases} \dfrac{\partial S}{\partial x'} = 2\sum\limits_{i=1}^{p} (\omega_i - M)\left(\dfrac{\partial \omega_i}{\partial x}\right) = 0, \\ \dfrac{\partial S}{\partial y'} = 2\sum\limits_{i=1}^{p} (\omega_i - M)\left(\dfrac{\partial \omega_i}{\partial y}\right) = 0, \\ \dfrac{\partial S}{\partial M} = 2\sum\limits_{i=1}^{p} (\omega_i - M)(-1) = 0. \end{cases} \tag{3.10}$$

Джерело [12,13]

Підставивши в (3.10) $\boldsymbol{\omega_i}$, i=1,2,...,p≥3 із (3.6), отримаємо три лінійних рівняння у вигляді:

$$\begin{cases} D_x x' + D_y y' + D_{\text{M}} M + D = 0, \\ G_x x' + G_y y' + G_M M + G = 0, \\ E_x x' + E_y y' + E_M M + E = 0, \end{cases} \tag{3.11}$$

Джерело [12,13]

де

$$D_x = \sum_{i=1}^{p} (\frac{A_i}{\beta_i})^2, \quad G_x = \sum_{i=1}^{p} (\frac{A_i B_i}{\beta_i^2}), \quad E_x = \sum_{i=1}^{p} (\frac{A_i}{\beta_i}),$$

$$D_y = \sum_{i=1}^{p} (\frac{A_i B_i}{\beta_i^2}), \quad G_y = \sum_{i=1}^{p} (\frac{B_i}{\beta_i})^2, \quad E_y = \sum_{i=1}^{p} (\frac{B_i}{\beta_i}),$$

$$D_M = -\sum_{i=1}^{p} (\frac{A_i}{\beta_i}), \quad G_M = -\sum_{i=1}^{p} (\frac{B_i}{\beta_i}), \quad E_M = -1,$$

$$D = \sum_{i=1}^{p} (\frac{A_i C_i}{\beta_i^2}), \quad G = \sum_{i=1}^{p} (\frac{B_i C_i}{\beta_i^2}), \quad E = \sum_{i=1}^{p} \frac{C_i}{\beta_i}.$$

Джерело [12,13]

Розв'язок системи (3.11) менш трудомістке, так як для розрахунку коефіцієнтів застосовується вже не подвійна сума, а одинарна.

Далі ще більше спростимо функціонал (3.9), задав масштабний множник М як константу, наприклад, М= 1. При цьому коефіцієнти ω_i автоматично врахують масштабне змінення. Отримаємо функціонал у вигляді:

$$S = \sum_{i=1}^{p} (\omega_i - 1)^2 \longrightarrow \min, \; p \geq 3. \qquad (3.12)$$

Джерело [12,13]

Продиференціювавши, отримаємо систему:

$$\begin{cases} \dfrac{\partial S}{\partial x} = 2\sum_{i=1}^{p} (\omega_i - 1)\left(\dfrac{\partial \omega_i}{\partial x}\right) = 0, \\ \dfrac{\partial S}{\partial y} = 2\sum_{i=1}^{p} (\omega_i - 1)\left(\dfrac{\partial \omega_i}{\partial y}\right) = 0. \end{cases} \qquad (3.13)$$

Джерело [12,13]

Підставивши в (3.12) ω_i, i=1,2,...,p≥3 из (3.6), отримаємо два лінійних рівняння у вигляді (3.8), де

$$D_x = \sum_{i=1}^{p} (\frac{A_i}{\beta_i})^2, \quad G_x = \sum_{i=1}^{p} (\frac{A_i B_i}{\beta_i^2}),$$
$$D_y = \sum_{i=1}^{p} (\frac{A_i B_i}{\beta_i^2}), \quad G_y = \sum_{i=1}^{p} (\frac{B_i}{\beta_i})^2, \qquad (3.14)$$
$$D = \sum_{i=1}^{p} (\frac{A_i C_i}{\beta_i^2}), \qquad G = \sum_{i=1}^{p} (\frac{B_i C_i}{\beta_i^2}).$$

Джерело [12,13]

Ці політканинні перетворення найбільш прості і найменш трудомісткі з обчислювальної точки зору.

Як показує досвід, ці всі три описані політканинні перетворення мало відрізняються одне від одного.

Далі в політканинні перетворення можна ввести додатково вагові коефіцієнти w_i, на які у функціоналах (3.5), (3.9), (3.12) будут помножені відповідно ω_i. При цьому і-а базова лінія, у якої буде більше значення w_i, буде більше впливати на перетворення точки. Геометрчно це буде виглядати таким чином, що ця базова лінія немовби притягує до себе точки об'єкта. На основі застосування вагових функцій можна різноманітнити варіанти перетворень, наприклад, вагові коефіцієнти розраховувати як функції відстанів до базових ліній:

$$w_i = d^k, \qquad (3.15)$$

Джерело [12,13]

де d – відстань до базової лінії,

k – задаваєма степінь відстані.

При k<0 точка буде більше залежати від зміни ближчої базової лінії і менше від подальшої.

3.5 Користувацький інтерфейс

Для розроблення інтерфейсу було використано пакет appjar. Appjar – це найпростіший спосіб створення графічних інтерфейсів в Python. Вхідні віджети використовуються для фіксаціїї взаємодій з користувачем, або натисканням, введенням чи перетягуванням. Зазвичай вони забезпечують три функції:

— ADD - створює віджет;

— GET - отримує результат / стан віджета;

— SET - змінює те, що знаходиться у віджеті.

Записи використовуються для фіксації набраного вводу від користувача. Існує п'ять видів записів:

— NumericEntry - це дозволяє вводити лише числа;

— SecretEntry – з’являться зірки замість введених букв - корисно для збору паролів;

— AutoEntry - має список слів для автоматичного заповнення;

— ValidationEntry - може бути встановлено як дійсне / недійсне / очікування - зафарбує межу зеленим / червоним / чорним та покаже ✓ / ✕ / ★;

— OpenEntry / SaveEntry / DirectoryEntry - забезпечує кнопку для вибору файлу / каталогу та автоматично заповнює запис.

Щоб створити додаток appJar потрібно імпортувати gui з бібліотеки appJar та створіть змінну gui(рис. 3.6). Для цього треба додати наступний рядок на початку файлу вихідного коду:

```
1  # import the library
2  from appJar import gui
3
4  # let app be name of gui variable
5  app = gui()
```

Рисунок 3.6 – Код для створення додатку

За допомогою змінної програми треба налаштувати зовнішній вигляд програми та логіку кожного віджета. Наприклад, тут створюється вікно, яке відображає "Hello World" за допомогою змінної програми.

```
7  app.addLabel("title", " Hello World! ")
8
9  app.setLabelBg("title", "blue")
```

Рисунок 3.7 – Код для створення вікна

Нарешті, потрібно запустити додаток, додавши у свій код наступну команду:

```
11 app.go()
12
```

Рисунок 3.8 – Команда app.go()

Скомпілюємо даний код та отримаємо результат:

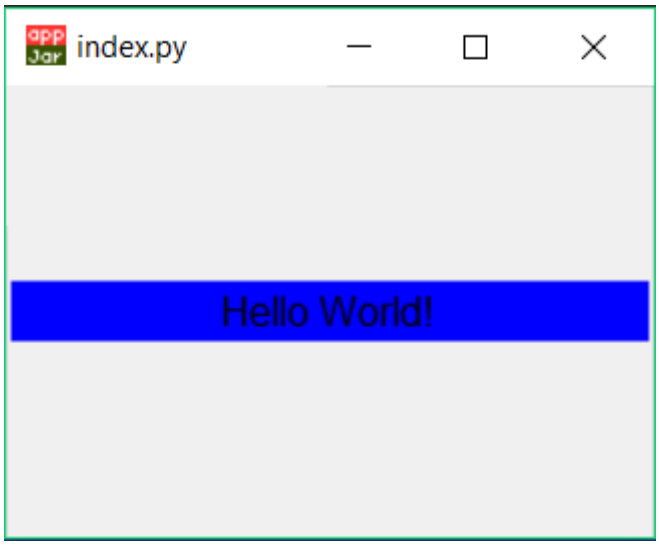

Рисунок 3.9 – Результат програми

Кнопка, яку можна натиснути, викликає функцію. Це ключ до запуску інтерактивного додатку. Графічний інтерфейс циклічно чекає, що щось станеться. Натискання кнопки - це класичний спосіб розпочати взаємодію з графічним інтерфейсом.

```
1   from appJar import gui
2
3   def press(btn):
4       print(btn)
5
6   app=gui()
7   app.addButton("One", press)
8   app.addButton("Two", press)
9   app.addButton("Three", press)
10  app.go()
```

Рисунок 3.10 – Код для створення кнопки

При запуску коду, отримаємо меню з трьома кнопками: one, two, three. При натисканні на одну з кнопок, спрацює функція press, тобто виведеться назва кнопки, яку натиснув користувач.

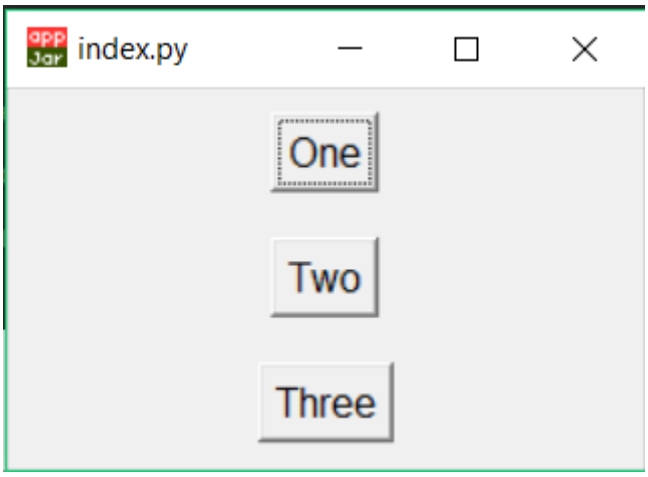

Рисунок 3.11 – Результат програми

Щоразу, коли будь-яка функція викликається GUI, заголовок віджета, який її викликав, передається як параметр. Таким чином, кілька віджетів можуть використовувати одну і ту ж функцію, але можна виконувати різні дії, залежно від імені, переданого як параметр.

Мітки використовуються для відображення тексту в графічному інтерфейсі. Вони чудово підходять для заголовків у верхній частині графічного інтерфейсу, як правило, охоплюють кілька стовпців. Вони дійсно корисні перед записами та випадами, щоб пояснити їх призначення. І вони дуже корисні внизу графічного інтерфейсу, щоб показати результати дії.

```python
from appJar import gui

app = gui()

app.addLabel("l1", "Label 1")
app.addLabel("l2", "Label 2")
app.addLabel("l3", "Label 3")
app.addLabel("l4", "Label 4")
# common set functions
app.setLabelBg("l1", "red")
app.setLabelBg("l2", "yellow")
app.setLabelBg("l3", "purple")
app.setLabelBg("l4", "orange")

app.go()
```

Рисунок 3.12 – Команда app.go()

Результат програми:

Рисунок 3.13 – Результат програми

3.5 Висновки до розділу

У даному розділі було наведено перелік інструментів, з допомогою який створювалася система прогнозування розвитку забруднень водяних(земних) поверхонь. Були використані переваги мови програмування Python. Також були

використан пакет Appjar, за допомогою якого було створено GUI. Для побудови графіка було використано пакет Matplotlib та його методи. Бібліотека Numpy була використана для обчислення системи рівнянь. А також був використан метод полікоординатних перетворень для реалізації прогнозування розвитку забруднень.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблена система має мати такий фунціонал:

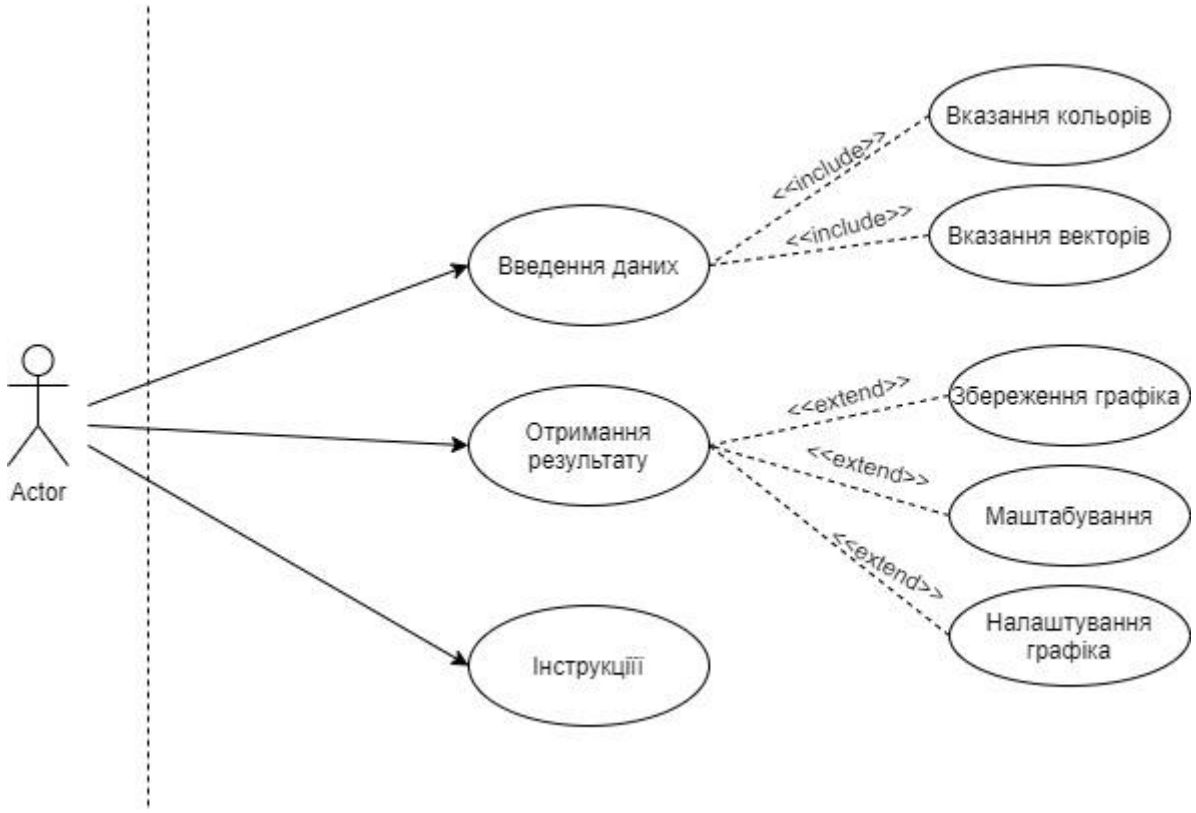

Рисунок 4.1 – Функціонал програмного модуля

При розробці системи спершу були підключені бібліотеки, описані в минулому розділі. За допомогою import код Python в одному модулі отримує доступ до коду в іншому модулі. Тобто коли модуль імпортується, Python виконує весь код у файлі

модуля. Також такі функції, як importlib.import_module () та вбудований __import __ (), можуть використовуватися для виклику та підключення пакетів або бібліотек.

Заява про імпорт поєднує дві операції; він шукає названий модуль, а потім прив'язує результати цього пошуку до імені в локальній області. Операція пошуку імпорту оператора визначається як виклик до функції __import __ () з відповідними аргументами. Повернене значення __import __ () використовується для виконання операції зв’язування імені оператора імпорту.

Прямий виклик __import __ () виконує лише пошук модуля та, якщо його знайдено, операцію створення модуля. Хоча можуть виникати певні побічні ефекти, такі як імпорт батьківських пакетів та оновлення різних кешів (включаючи sys.modules), лише оператор імпорту виконує операцію зв’язування імені.

Коли виконується оператор імпорту, викликається стандартна вбудована функція __import __ (). Інші механізми виклику системи імпорту (наприклад, importlib.import_module ()) можуть вибрати обхід __import __ () та використовувати власні рішення для реалізації семантики імпорту.

Коли модуль вперше імпортується, Python шукає модуль, і якщо його знайдено, він створює модульний об'єкт, ініціалізуючи його. Якщо названого модуля неможливо знайти, піднімається ModuleNotFoundError. Python реалізує різні стратегії пошуку названого модуля, коли викликається import. Ці стратегії можна змінювати та розширювати, використовуючи різні способи.

Для зручності використання бібліотек іноді використовується конструкція import … as …. Тобто замість назви бібліотеки при використанні її, в коді буде написана не повна назва, а ім’я, яке надасть цій бібліотеці розробник. В моєму випадку пакет math я переіменовую в букву m, а matplotlib.pyplot у plt(рисунок 4.2).

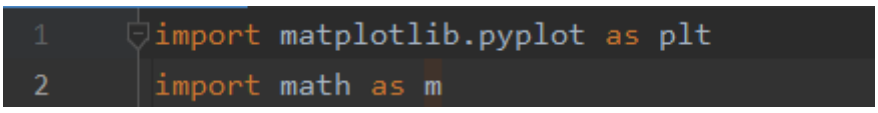

Рисунок 4.2 – Використання import … as …

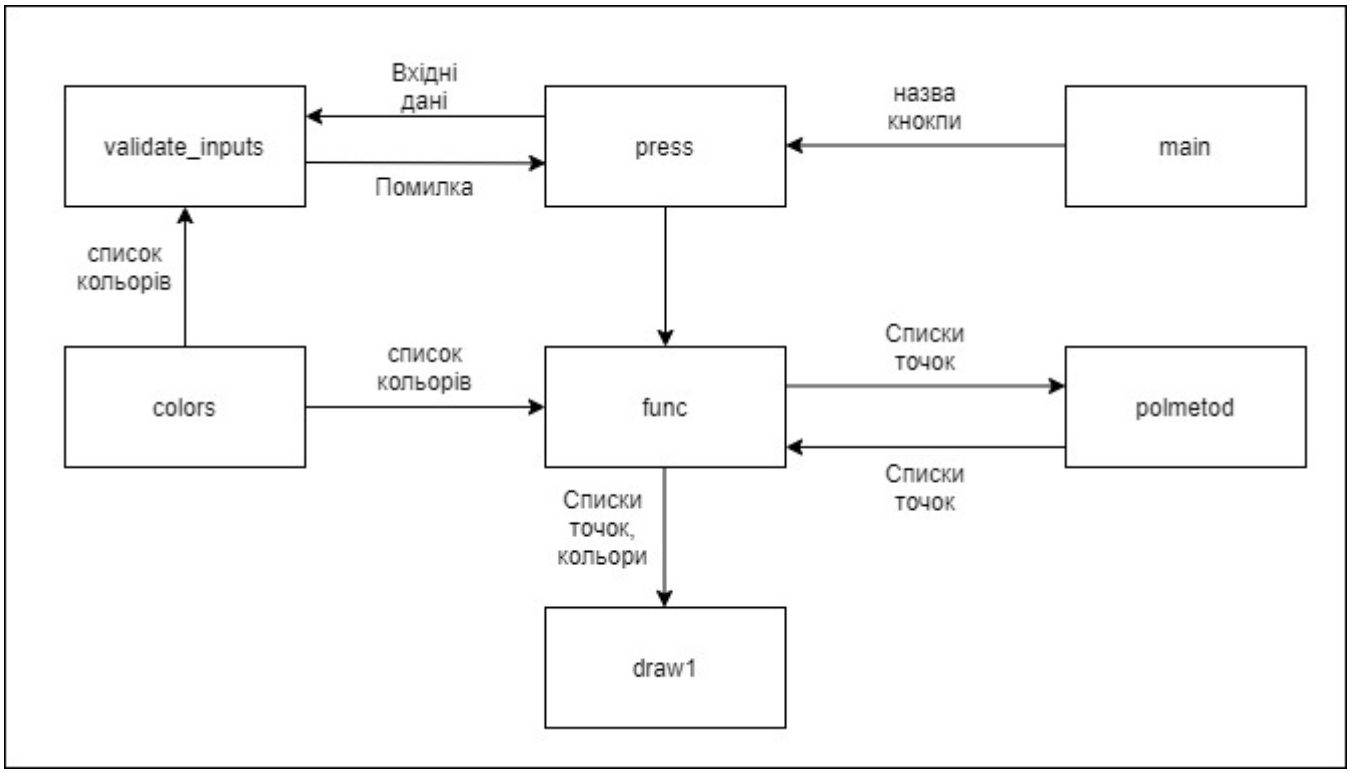

Рисунок 4.3 – Діаграма

Після підключення пакетів йдуть функції, в яких створюється інтерфейс, будується графік та відбуваються обчислення. Функція - це блок коду, який запускається лише тоді, коли він викликається. Можна передавати дані, тобто параметри, у функції. Також функція може повертати дані.

В Python функція визначається за допомогою ключового слова def (рисунок 4.3).

```
1    def my_function():
2      print("Hello from a function")
```

Рисунок 4.4 – Використання def

Щоб викликати функцію потрібно використовувати ім'я функції з круглими дужками:

```
1  def my_function():
2      print("Hello from a function")
3
4  my_function()
```

Рисунок 4.5 – Виклик функції

Інформація може передаватися у функції як аргументи. Аргументи задаються після назви функції, усередині дужок. Можна додати скільки завгодно аргументів, просто розділивши їх комою.

Останніми рядками коду є виклик головної функції main. Основна функція - точка входу будь-якої програми. Але інтерпретатор python виконує код вихідного файлу послідовно і не викликає жодного методу, якщо він не входить до коду. Але якщо це безпосередньо частина коду, він буде виконуватися, коли файл імпортується як модуль.

Існує спеціальна методика визначення основного методу в програмі python, щоб він виконувався лише тоді, коли програма запускається безпосередньо, а не виконується при імпорті в якості модуля. Приклад визначання головної функції python у простій програмі.

```
1   print("Hello")
2
3   print("__name__ value: ", __name__)
4
5
6   def main():
7       print("python main function")
8
9
10  if __name__ == '__main__':
11      main()
```

Рисунок 4.6 – Визначення головної функції

Коли програма Python виконується, інтерпретатор починає виконувати код

всередині неї. Він також встановлює кілька неявних змінних значень, одне з них - __name__, значення якого встановлено як __main__. Для основної функції python, потрібно визначити функцію, а потім використовувати, якщо __name__ == '__main__', умову для виконання цієї функції. Якщо вихідний файл Python імпортується як модуль, інтерпретатор встановлює значення __name__ на ім'я модуля, тому умова if поверне помилку і основний метод не буде виконаний. Python надає нам можливість зберігати будь-яке ім’я для основного методу, однак найкращою практикою називати його як метод main ().

Для перевірки введення даних користувачем була створена функція, за допомогою якої відбувається валідація полей і виводиться помилка, якщо дані не відповідають дійсності.

```
def validate_inputs(r,r1,color1,color2, num_st):
```

Рисунок 4.7 – Функція validate_inputs

Дана функція має назву validate_inputs. Вона приймає п’ять параметрів r,r1,color1,color2, num_st, тобто радіус кола, радіус многокутника, колір кола, колір многокутника та кількість кроків відповідно. Користувач може ввести лише числове значення в параметри r , r1 та num_st. Тобто якщо користувач буде вводити у дані поля все, окрім цифр, то ці дані вводитися не будуть. Також є перевірка на введення тільки додатніх чисел у параметри r, r1 та num_st, тобто користувач зможе ввести від’ємні числа у строку, але при запуску програми виникне помилка, яка повідомить про те, що потрібно ввести інше число.

За умовою радіус многокутника повинен бути більшим, ніж радіус кола, тому існує перевірка, яка порівнює дані числа та виводить помилку, якщо радіус многокутника менший, ніж радіус кола.

```
178     if r1 <= r:
179         errors = True
180         error_msgs.append("Please enter a valid radius of polygon")
```

Рисунок 4.8 – Перевірка даних

Для параметрів color та color1 також існують перевірки. Для коректного вводу даних параметрів була створена функція colors, яка повертає список усіх кольорів.

```
def colors():
    cnames = ['aliceblue', 'antiquewhite', 'aqua', 'aquamarine', 'azure', 'beige', 'bisque',
              'black', 'blanchedalmond', 'blue', 'blueviolet', 'brown', 'burlywood', 'cadetblue',
              'chartreuse', 'chocolate', 'coral', 'cornflowerblue', 'cornsilk', 'crimson', 'cyan', 'darkblue',
              'darkcyan', 'darkgoldenrod', 'darkgray', 'darkgreen', 'darkkhaki', 'darkmagenta', 'darkolivegreen',
              'darkorange', 'darkorchid', 'darkred', 'darksalmon', 'darkseagreen', 'darkslateblue', 'darkslategray',
              'darkturquoise', 'darkviolet', 'deeppink', 'deepskyblue', 'dimgray', 'dodgerblue', 'firebrick',
              'floralwhite', 'forestgreen', 'fuchsia', 'gainsboro', 'ghostwhite', 'gold', 'goldenrod', 'gray',
              'green', 'greenyellow', 'honeydew', 'hotpink', 'indianred', 'indigo', 'ivory', 'khaki', 'lavender',
              'lavenderblush', 'lawngreen', 'lemonchiffon', 'lightblue', 'lightcoral', 'lightcyan', 'lightgoldenrodyellow',
              'lightgreen', 'lightgray', 'lightpink', 'lightsalmon', 'lightseagreen', 'lightskyblue', 'lightslategray',
              'lightsteelblue', 'lightyellow', 'lime', 'limegreen', 'linen', 'magenta', 'maroon', 'mediumaquamarine', 'mediumblue',
              'mediumorchid', 'mediumpurple', 'mediumseagreen', 'mediumslateblue', 'mediumspringgreen', 'mediumturquoise',
              'mediumvioletred', 'midnightblue', 'mintcream', 'mistyrose', 'moccasin', 'navajowhite', 'navy', 'oldlace', 'olive',
              'olivedrab', 'orange', 'orangered', 'orchid', 'palegoldenrod', 'palegreen', 'paleturquoise', 'palevioletred',
              'papayawhip', 'peachpuff', 'peru', 'pink', 'plum', 'powderblue', 'purple', 'red', 'rosybrown', 'royalblue',
              'saddlebrown', 'salmon', 'sandybrown', 'seagreen', 'seashell', 'sienna', 'silver', 'skyblue', 'slateblue', 'slategray',
              'snow', 'springgreen', 'steelblue', 'tan', 'teal', 'thistle', 'tomato', 'turquoise', 'violet', 'wheat', 'white',
              'whitesmoke', 'yellow', 'yellowgreen']
    return cnames
```

Рисунок 4.9 – Функція, яка повертає список кольорів

За допомогою конструкції if … not in … я шукаю колір зі списку, який співпадає із введеними даними у параметрах color та color1.
Оцінюється як true, якщо він не знаходить змінної у зазначеній послідовності, а false - в іншому випадку. Якщо введеного кольора не існує, то користувач отримає помилку.

За для швидшого заповнення полей, де потрібно ввести кольори фігур, існує функція автозаповнення, для якої використовується та ж сама функція colors, яка повертає список кольорів.

```
words = colors()
app.addAutoEntry("Color of circle: ", words)
app.setAutoEntryNumRows("Color of circle: ", 4)
```

Рисунок 4.10 – Функція автозаповненя

Спочатку повертається список кольорів та записується у змінну words, далі за допомогою пакета Appjar, визивається функція addAutoEntry, де вказується список з кольорами. В наступній функції вказується кількість стовпчиків, які показуються користувачеві при вводі даних у поля. Це дозволяє користувачу швидко ввести дані

без помилок.

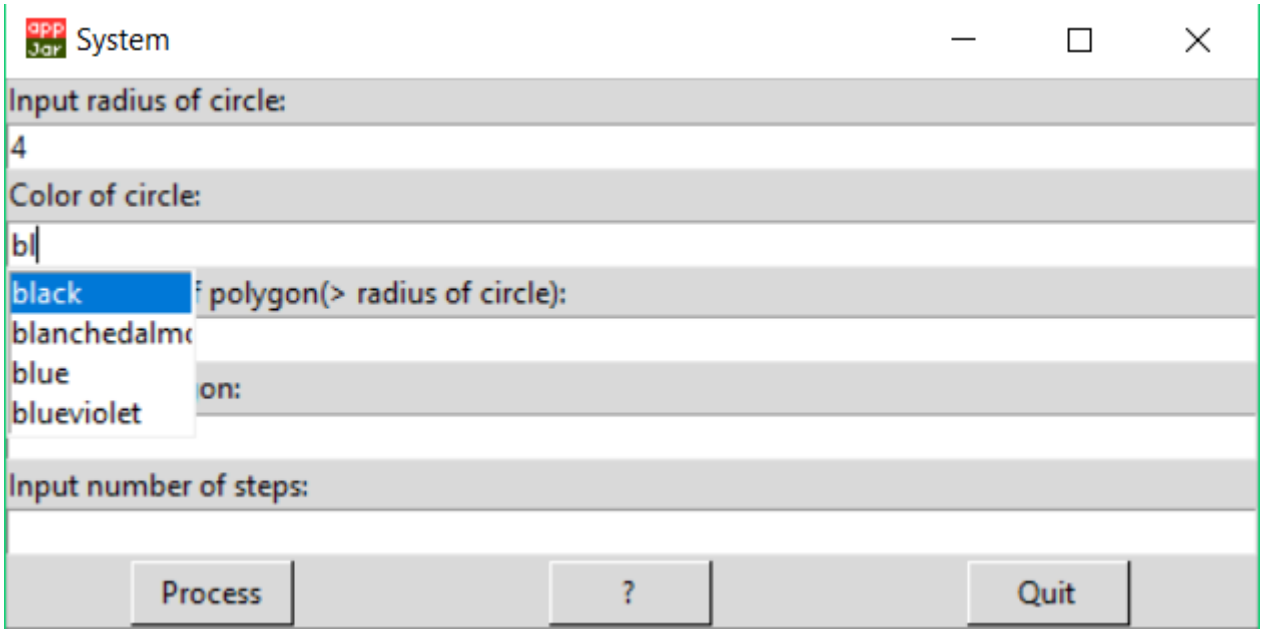

Рисунок 4.11 – Автозаповнення

Якщо у користувача не виходить правильно ввести дані, то існує кнопка «?», де написані інструкції для успішного запуску програми.

Для обробки кнопок була створена функція press, яка отримує параметр button. Тобто при натисканні на одну із кнопок, її назва передається у дану функцію.

Далі за допомогою умовного оператора if, передана назва обробляється і вибирає, яку дію слід виконати, в залежності від значення змінних в момент перевірки умови.

Натиснувши кнопку Process, програма отримає введені дані, викличе функцію для перевірки даних. Якщо будуть помилки, то повідомить про це. Якщо помилок немає, то програма викличе функцію func, де відбуваються усі обчислення для отримання кінцевого результату. Якщо користувач натисне кнопку ? , то програма викличе .infoBox і повідомить користувачу про інструкції. В іншому випадку, тобто при натисканні кнопки Quit, програма завершить роботу.

```
def press(button):

    if button == "Process":
        r = app.getEntry('Input radius of circle: ')
        r1 = app.getEntry('Input radius of polygon(> radius of circle): ')
        color1 = app.getEntry("Color of circle: ")

        color2 = app.getEntry("Color of polygon: ")
        num_st = app.getEntry('Input number of steps: ')

        errors, error_msg = validate_inputs(r,r1, color1, color2, num_st)
        if errors:
            app.errorBox("Error", "\n".join(error_msg), parent=None)
        else:
            func(r,r1, color1, color2,num_st)
    elif button == "?":
        app.infoBox("Instructions", "1. Enter radius of circle >0 \n2. Enter color from dropbox \n3. Enter radius of rectangle > radius of circle \
                                    "4. Enter number of ssteps > 0 \n5. Click Process \n6. Click Quit to exit from the program", parent=None)
    else:
        app.stop()
```

Рисунок 4.12 – Визначення функції press

В найголовнішій функції func відбуваються обробка даних, обчислення за формулами та вивід кінцевого результату. Були використані списки,
Список є найбільш універсальним типом даних, доступним у Python, який може бути записаний у вигляді списку розділених комами значень (елементів) між квадратними дужками. Важливим у списку є те, що елементи в списку не повинні бути одного типу.

Створення списку так само просто, як і розміщення різних розділених комами значень між квадратними дужками. Наприклад:

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed data types
my_list = [1, "Hello", 3.4]
```

Рисунок 4.13 – Код для створення списків

4.1 Висновки

В даному розділі було описано програмну реалізацію системи розвитку забруднень водяних(земних) поверхонь. Програму реалізовано на мові

програмування Python. Були використані різні бібліотеки, а сама програма була реалізована за допомогою функцій. Графік було побудовано за допомогою полікоординатних перетворень. Інтерфейс користувача реалізований за допомогою бібліотеки Appjar.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

5.1 Запуск системи

Для запуску розробленої програми у форматі .py необхідно:

1. Скачати IDLE з офіційного сайту.
2. Запустити програму, натиснувши на неї або через командний рядок, вказавши назву файлу.

Для запуску програми у форматі .exe потрібно всього лише два рази клікнути на неї.

5.2 Інтерфейс користувача

При запуску системи користувачу випадає меню, в якому потрібно вказати деякі параметри (рисунок 5.1).

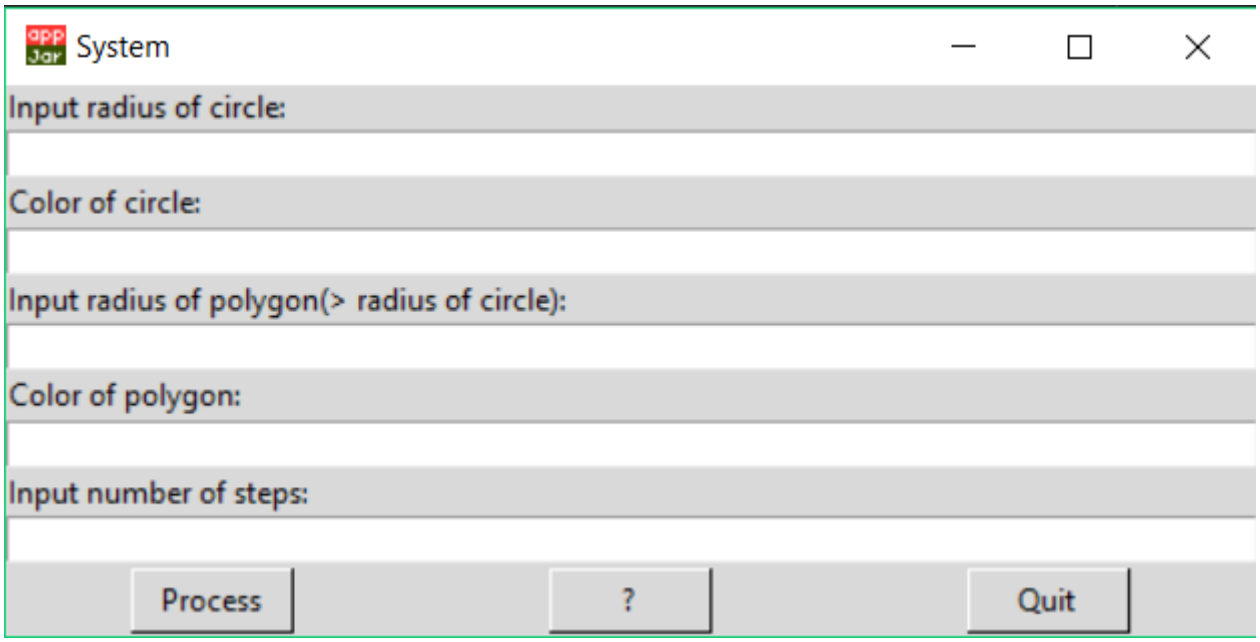

Рисунок 5.1 – Меню

Після вводу параметрів в полях, вікно має буде таким (рисунок 5.2).

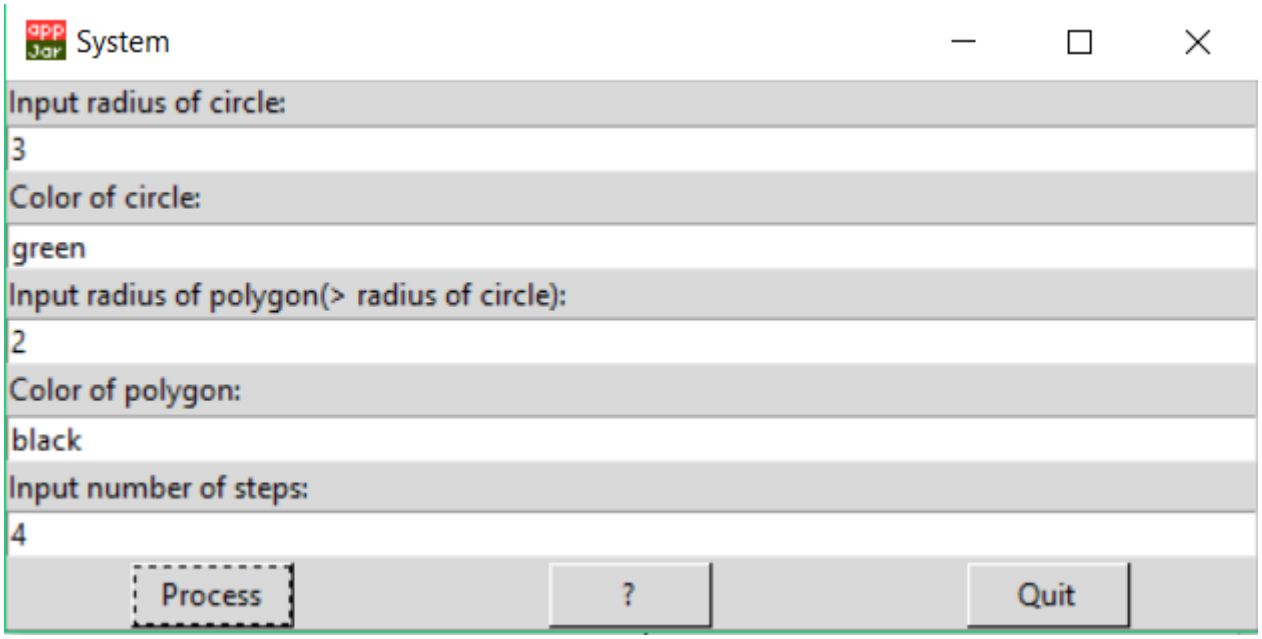

Рисунок 5.2 – вікно з даними у полях

Після того як користувач натисне кнопку Process під формами вводу, у разі успішного вводу даних, відкриється нове вікно з результатом програми(рисунок 5.2).

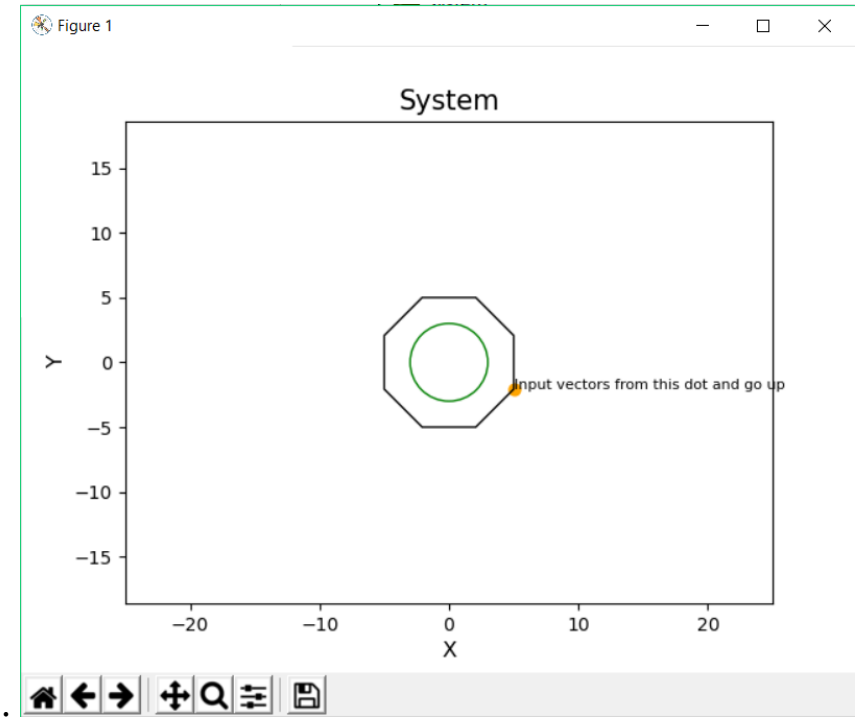

Рисунок 5.3 – Результат програми

У перше поле потрібно ввести радіус кола. Якщо користувач захоче ввести в це поле текст, то він вводитись не буде, бо у першому полі та у третьому, де потрібно ввести радіус кола та багатокутника, є валідація поля, тому там за замовчуванням можливо ввести лише число. При успішному введені даних, програма буде працювати.

У третьому полі потрібно ввести радіус багатокутника. За замовчуванням він повинен бути більшим, ніж радіус кола, тому якщо користувач введе менший радіус, то з’явиться помилка(рисунок 5.4).

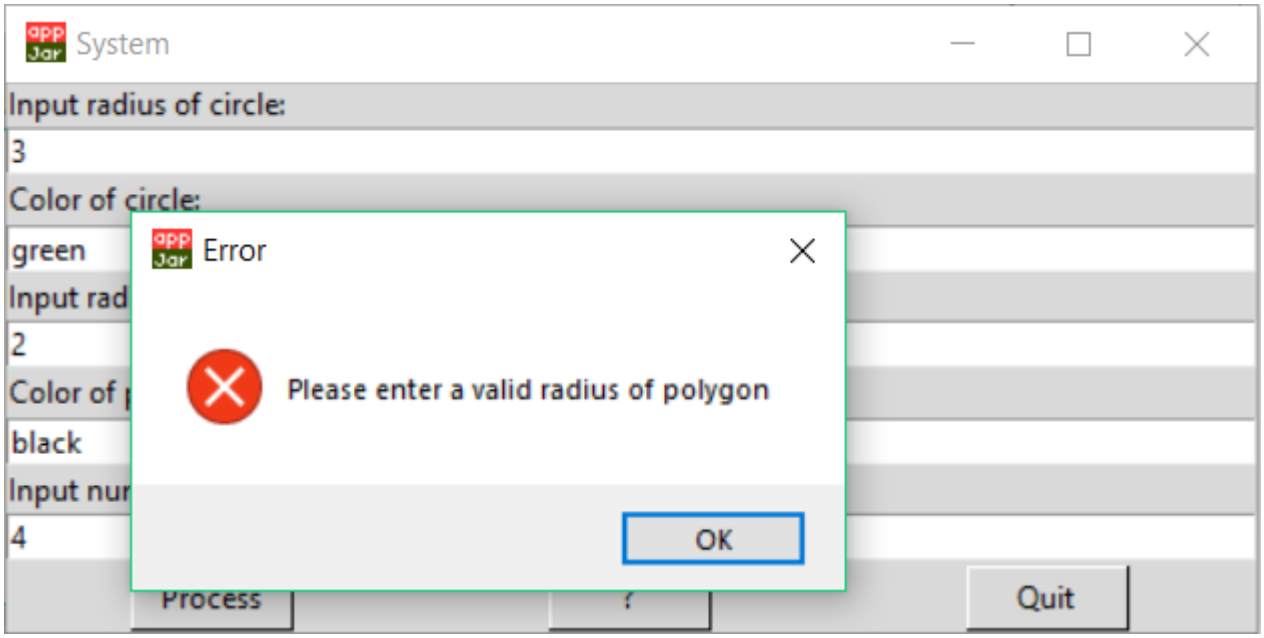

Рисунок 5.4 – Помилка введення радіусу багатокутника

У другому полі та у четвертому, де потрібно ввести колір кола та багатокутника, є перевірка на введення даних. Якщо буде введено будь-який текст або числа, то програма при натиску кнопки Process, повідомить про помилку(рисунок 5.5). В дані поля потрібно вводити назву кольора на англійській мові. Якщо введені дані будуть правильними, то програма буде працювати.

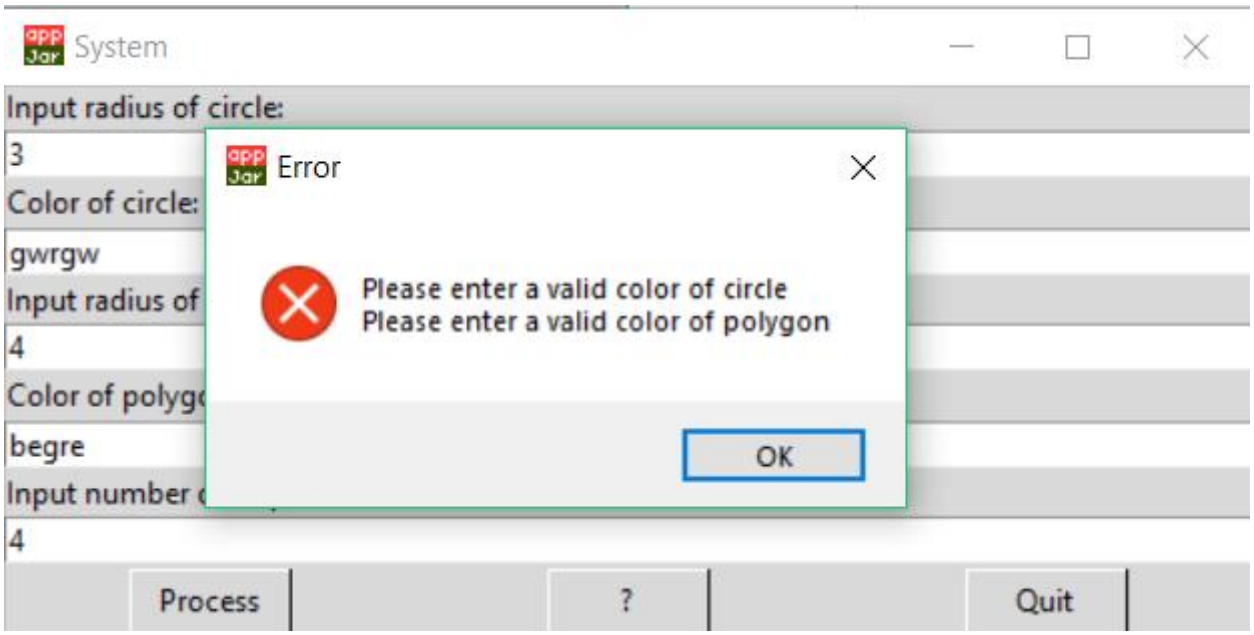

Рисунок 5.5 – Помилка введення кольорів кола та багатокутника

Після того як користувач успішно введе дані та натисне кнопку Process, з’явиться нове вікно з побудованим колом та багатокутником(рисунок 5.6). У цьому вікні потрібно вказати вектори багатокутника, починаючи з виділеної точки та рухаючись вгору по номерам. Користувачу потрібно натиснути на графік біля вказаних точок. Після чого на графіку будуть поставлені червоні точки(рисунок 5.7). З точок многокутника до вказаних точок будуть проведені вектори(рисунок 5.8).

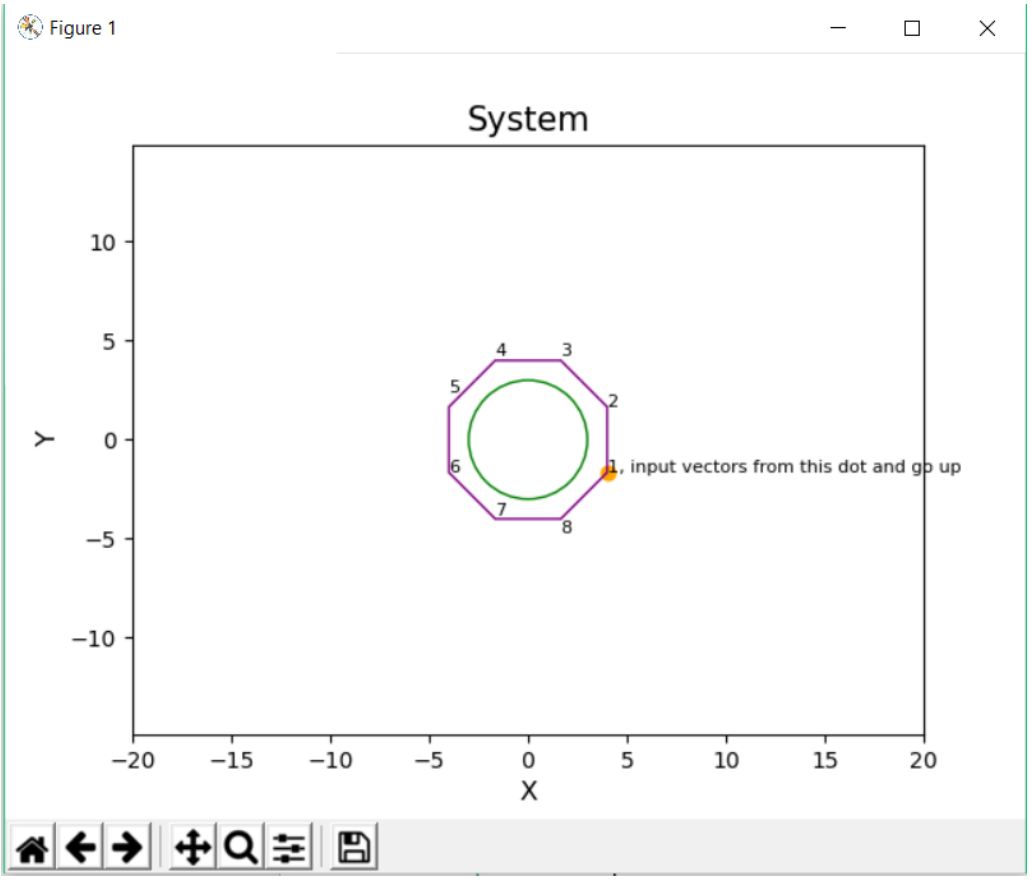

Рисунок 5.6 – Вікно з побудованим колом та багатокутником

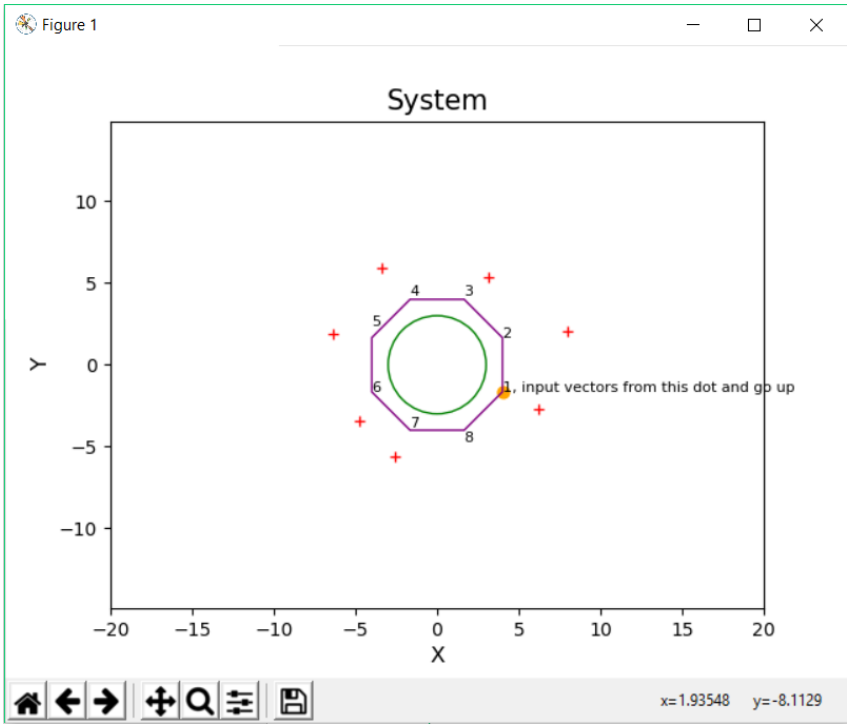

Рисунок 5.7 – Введення точок для побудови векторів

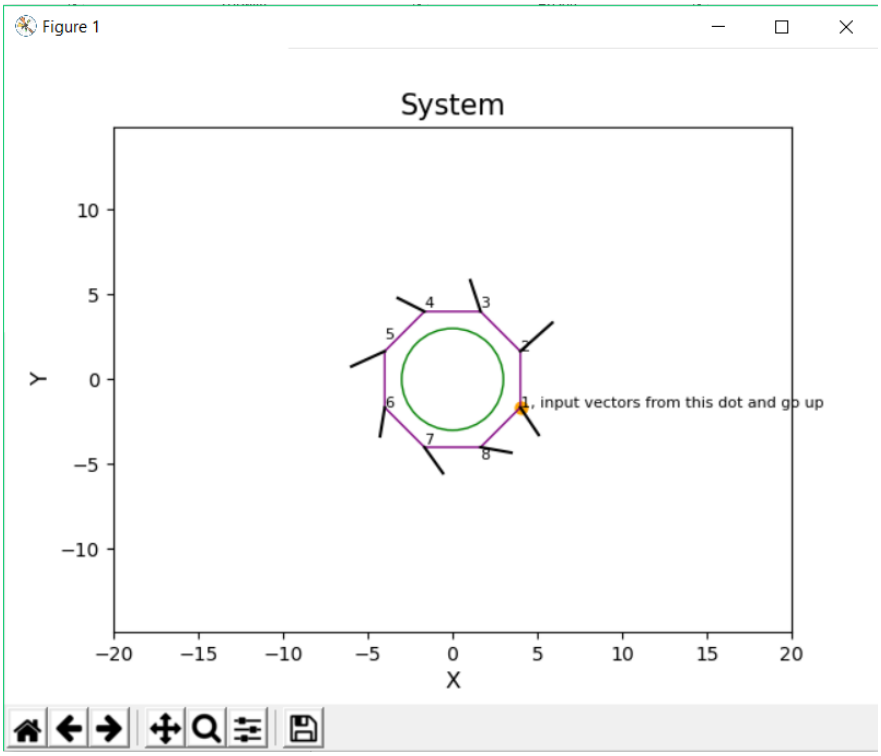

Рисунок 5.8 – побудовані вектори

Після успішного введення векторів, на даному графіку з’явиться кінцевий результат(рис.5.9).

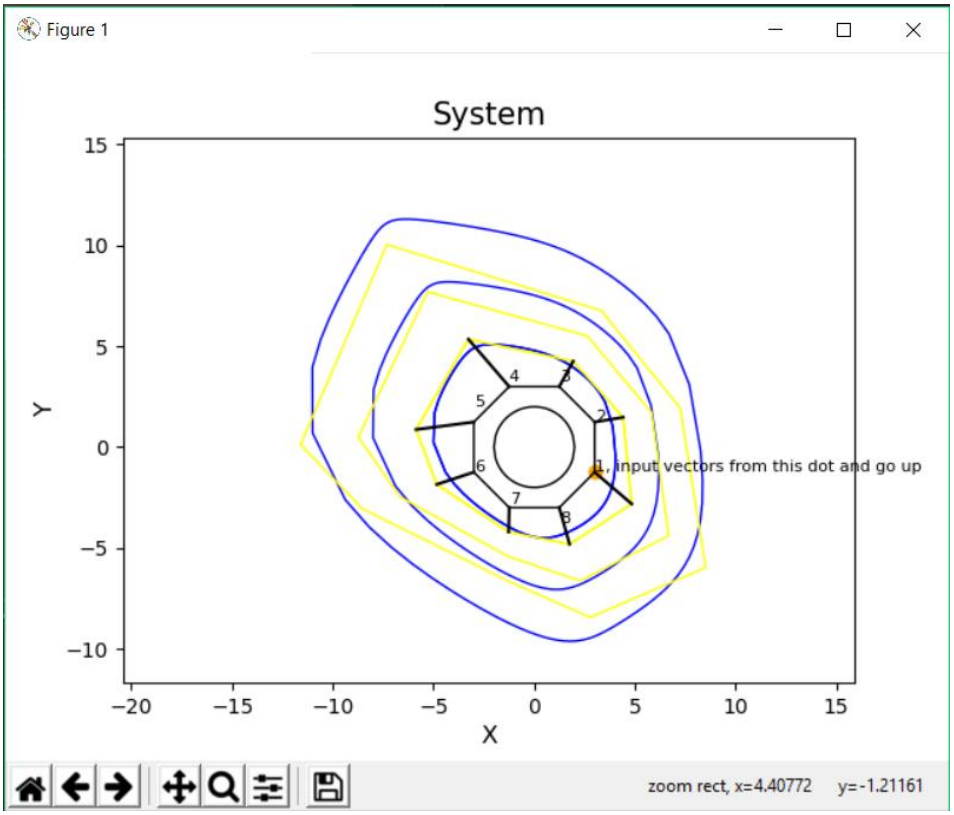

Рисунок 5.9 – Результат програми

Якщо користувач не зможе правильно запустити програму, то є кнопка зі знаком питання(рис. 5.10). Натиснувши на неї, відкриється вікно з усіми інструкціями для успішного запуску програми(рис. 5.11).

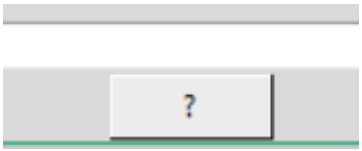

Рисунок 5.10 – Кнопка з інструкціями

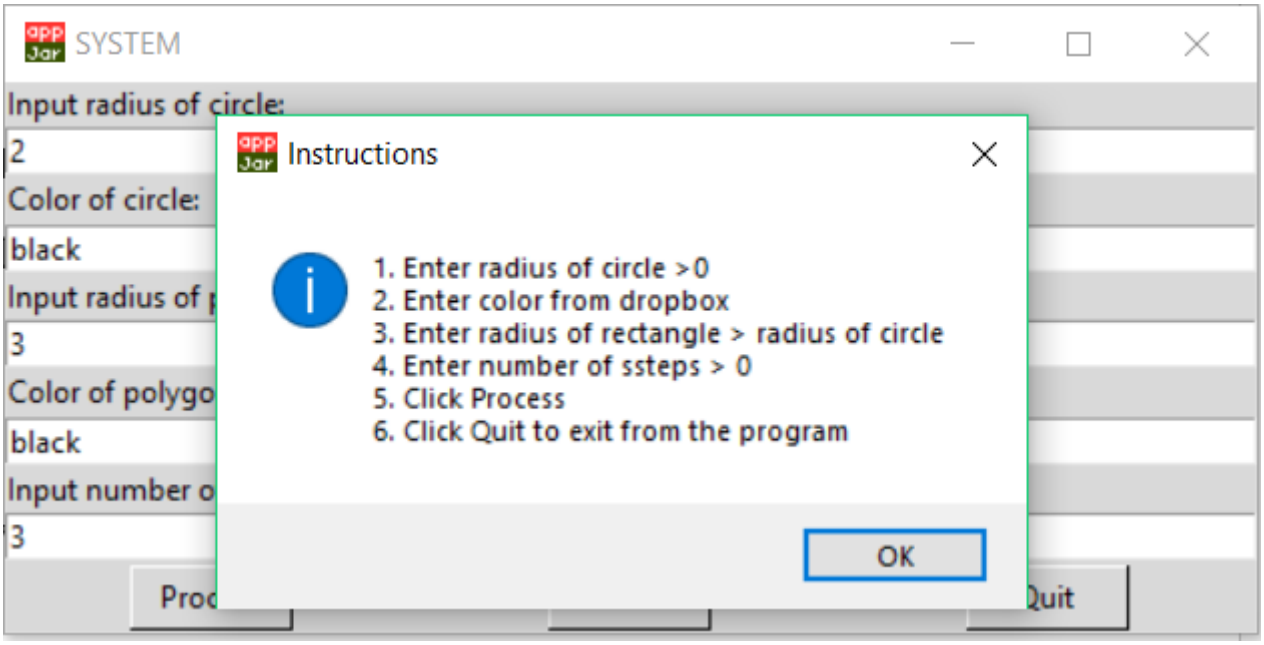

Рисунок 5.11 – Вікно з інструкціями

5.3 Висновки

У даному розділі було описано необхідні дії для запуску системи та наведено інтерфейс користувача. Пояснено як користуватися даною системою. Як приклад зроблено запуск програми, варіанти введення даних. Показано приклади помилок, що можуть виникнути під час роботи.

ВИСНОВКИ

Під час створення програмного модулю було покращено навички розробки програмних продуктів за допомогою середовища розробки програмного забезпечення PyCharm та мови програмування Python.

Також були отримані навички застосування бібліотек Python, які надають можливість у найкоротший час створити складні графіки. За допомогою бібліотек використаних у дипломному проекті, я навчився розробляти зручний та зрозумілий інтерфейс всього за декілька рядків коду.

В результаті роботи створено програмний продукт, який використовуючи переваги мови програмування Python та полікоординатний метод, дозволяє швидко отримувати потрібний результат.

Таким чином, розроблення даної програми поглибило знання різноманітних бібліотек Python, використання яких є невід'ємною частиною сучасних програмних продуктів. А також було проведено аналіз існуючих методів для прогнозування розвитку забруднень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.
2. What is Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/python_intro.asp.
3. What is Pycharm [Електронний ресурс] – Режим доступу до ресурсу: https://intellipaat.com/blog/what-is-pycharm/.
4. Matplotlib module [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/index.html.
5. Pyplot in matplotlib [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/pyplot-in-matplotlib/.
6. Math module [Електронний ресурс] – Режим доступу до ресурсу: https://realpython.com/python-math-module/.
7. Appjar module [Електронний ресурс] – Режим доступу до ресурсу: http://appjar.info/.
8. NumPy module [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/numpy_intro.asp.
9. Графічне моделювання – Бадаєв Ю.І.
10. Бадаєв Ю.І. Геометричне моделювання криволінійних обводів складних об’єктів.-К.:ТЕХТ,2015,-172с.
11. Ю.И. Бадаев. Поликоординатный метод в прикладной геометрии и компьютерной графике.-К.: Просвіта, 2006.-173с.
12. Import module [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/reference/import.html.
13. Python functions [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/python_functions.asp.

14. Python main function [Електронний ресурс] – Режим доступу до ресурсу: https://www.journaldev.com/17752/python-main-function.
15. Python lists [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/python/python_lists.html.

Додаток А

Система прогнозування розвитку забруднень(водяних) земних поверхонь

УКР.НТУУ“КПІ”.ТВ6145_20Б

Специфікація

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ“КПІ”.ТВ6145_ 20Б	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ».ТВ6145_ 20Б 12-1	Текст програмного модулю	
УКР.НТУУ«КПІ».ТВ6145_ 20Б 13-1	Опис програмного модулю	

Додаток Б

Система прогнозування розвитку забруднень водяних(земних) поверхонь

УКР.НТУУ«КПІ».ТВ6145_20Б 12-1

Текст програмного модулю

Аркушів 7

2020

```
#Підключення бібліотек
import matplotlib.pyplot as plt
import math as m
import matplotlib.patches
import matplotlib.lines
import matplotlib.path
import numpy as np
from appJar import gui

#Основна функція main
def main():
    global r, r1, circle1, polygon1, a, b,app, color1, color2
    app = gui("SYSTEM", useTtk=True)
    app.setTtkTheme("default")
    app.setSize(500, 200)
    # Додавання інтерактивних компонентів
    app.addLabel('Input radius of circle: ')
    app.addNumericEntry('Input radius of circle: ')
    app.addLabel("Color of circle: ")
    words = colors()
    app.addAutoEntry("Color of circle: ", words)
    app.setAutoEntryNumRows("Color of circle: ", 4)
    app.addLabel('Input radius of polygon(> radius of circle): ')
    app.addNumericEntry('Input radius of polygon(> radius of circle): ')
    app.addLabel("Color of polygon: ")
    app.addAutoEntry("Color of polygon: ", words)
    app.setAutoEntryNumRows("Color of polygon: ", 3)
    app.addLabel('Input number of steps: ')
    app.addNumericEntry('Input number of steps: ')
    # Св'язування кнокпи с функцією під назвою press
```

```
    app.addButtons(["Process","?", "Quit"], press)
    # Запуск інтерфейса
    app.go()

#Полікоординатний метод
def polmetod(l,list1,listp,ax):
    listotvet = []
    listk = []

    for i in range(0, 8):
        A = l[i]
        B = l[i + 1]
        xa = A[0]
        ya = A[1]
        xb = B[0]
        yb = B[1]
        a = ya - yb
        b = xb - xa
        c = -(a * xa + b * ya)
        listk.append([a, b, c])

    for i in range(0, 100):
        At = list1[i]
        xa = At[0]
        ya = At[1]
        listdm = []
        Dx1 = 0
        Dy1 = 0
        Dt1 = 0
        Gx1 = 0
```

```
Gy1 = 0
Gt1 = 0
for j in range(0, 8):
    Am = listp[j]
    Bm = listp[j + 1]
    xam = Am[0]
    yam = Am[1]
    xbm = Bm[0]
    ybm = Bm[1]
    a = yam - ybm
    b = xbm - xam
    c = -(a * xam + b * yam)
    dm1 = a * xa + b * ya + c
    if dm1 < 0:
        dm1 = -1 * dm1
    dm = dm1 / (m.sqrt(a ** 2 + b ** 2))
    listdm.append(dm)
for i in range(0, 8):
    lw = listk[i]
    Aw = lw[0]
    Bw = lw[1]
    Cw = lw[2]
    Dx = (Aw / listdm[i]) ** 2
    Dx1 += Dx
    Dy = (Aw * Bw) / (listdm[i] ** 2)
    Dy1 += Dy
    Dt = (Aw * Cw) / (listdm[i] ** 2)
    Dt1 += Dt
    Gx = (Aw * Bw) / (listdm[i] ** 2)
    Gx1 += Gx
```

```
            Gy = (Bw / listdm[i]) ** 2
            Gy1 += Gy
            Gt = (Bw * Cw) / (listdm[i] ** 2)
            Gt1 += Gt

        M3 = np.array([[Dx1, Dy1], [Gx1, Gy1]])  # Матрица (левая часть системы)
        V3 = np.array([-Dt1, -Gt1])  # Вектор (правая часть системы)
        otvet = np.linalg.solve(M3, V3)
        H = otvet[0]
        Z = otvet[1]
        listotvet.append([H, Z])

    # Побудова результату
    polygonotvet = plt.Polygon(listotvet, color="blue", fill=False)
    ax.add_patch(polygonotvet)
    return listotvet

#Функція для перевірки даних
def validate_inputs(r,r1,color1,color2, num_st):
    cnames = colors()
    errors = False
    error_msgs = []
    if r <= 0:
        errors = True
        error_msgs.append("Please enter a valid radius of circle")

    if r1 <= r:
        errors = True
        error_msgs.append("Please enter a valid radius of polygon")
```

```
    if color1  not in cnames:
        errors = True
        error_msgs.append("Please enter a valid color of circle")

    if color2  not in cnames:
        errors = True
        error_msgs.append("Please enter a valid color of polygon")

    if num_st <= 0:
        errors = True
        error_msgs.append("Please enter a valid number of steps")

    return (errors, error_msgs)

#Функція обробки кнопок
def press(button):

    if button == "Process":
        r = app.getEntry('Input radius of circle: ')
        r1 = app.getEntry('Input radius of polygon(> radius of circle): ')
        color1 = app.getEntry("Color of circle: ")

        color2 = app.getEntry("Color of polygon: ")
        num_st = app.getEntry('Input number of steps: ')

        errors, error_msg = validate_inputs(r,r1, color1, color2, num_st)
        if errors:
            app.errorBox("Error", "\n".join(error_msg), parent=None)
        else:
            func(r,r1, color1, color2,num_st)
```

```
    elif button == "?":
        app.infoBox("Instructions", "1. Enter radius of circle >0 \n2. Enter color from
dropbox \n3. Enter radius of rectangle > radius of circle \n"
                                "4. Enter number of ssteps > 0 \n5. Click Process \n6. Click Quit to
exit from the program", parent=None)
    else:
        app.stop()

#Функція визначення кольорів
def colors():
    cnames = ['aliceblue', 'antiquewhite', 'aqua', 'aquamarine', 'azure', 'beige', 'bisque',
              'black', 'blanchedalmond', 'blue', 'blueviolet', 'brown', 'burlywood', 'cadetblue',
              'chartreuse', 'chocolate', 'coral', 'cornflowerblue', 'cornsilk', 'crimson', 'cyan',
'darkblue',
              'darkcyan', 'darkgoldenrod', 'darkgray', 'darkgreen', 'darkkhaki', 'darkmagenta',
'darkolivegreen',
              'darkorange', 'darkorchid', 'darkred', 'darksalmon', 'darkseagreen',
'darkslateblue', 'darkslategray',
              'darkturquoise', 'darkviolet', 'deeppink', 'deepskyblue', 'dimgray', 'dodgerblue',
'firebrick',
              'floralwhite', 'forestgreen', 'fuchsia', 'gainsboro', 'ghostwhite', 'gold', 'goldenrod',
'gray',
              'green', 'greenyellow', 'honeydew', 'hotpink', 'indianred', 'indigo', 'ivory', 'khaki',
'lavender',
              'lavenderblush', 'lawngreen', 'lemonchiffon', 'lightblue', 'lightcoral', 'lightcyan',
'lightgoldenrodyellow',
              'lightgreen', 'lightgray', 'lightpink', 'lightsalmon', 'lightseagreen', 'lightskyblue',
'lightslategray',
              'lightsteelblue', 'lightyellow', 'lime', 'limegreen', 'linen', 'magenta', 'maroon',
'mediumaquamarine', 'mediumblue',
```

```
            'mediumorchid', 'mediumpurple', 'mediumseagreen', 'mediumslateblue',
'mediumspringgreen', 'mediumturquoise',
            'mediumvioletred', 'midnightblue', 'mintcream', 'mistyrose', 'moccasin',
'navajowhite', 'navy', 'oldlace', 'olive',
            'olivedrab', 'orange', 'orangered', 'orchid', 'palegoldenrod', 'palegreen',
'paleturquoise', 'palevioletred',
            'papayawhip', 'peachpuff', 'peru', 'pink', 'plum', 'powderblue', 'purple', 'red',
'rosybrown', 'royalblue',
            'saddlebrown', 'salmon', 'sandybrown', 'seagreen', 'seashell', 'sienna', 'silver',
'skyblue', 'slateblue', 'slategray',
            'snow', 'springgreen', 'steelblue', 'tan', 'teal', 'thistle', 'tomato', 'turquoise', 'violet',
'wheat', 'white',
            'whitesmoke', 'yellow', 'yellowgreen']
    return cnames

#Побудова восьмикутника та кола
def draw1(list1,listp,color2,color1,ax):

    polygon1 = plt.Polygon(listp, color=color2, fill=False)
    polygonb = plt.Polygon(list1, color=color1, fill=False)
    ax.add_patch(polygon1)
    ax.add_patch(polygonb)
```

Додаток В

Система прогнозування розвитку забруднень водяних(земних) поверхонь

УКР.НТУУ«КПІ».ТВ6145_20Б 13-1

Опис програмного модулю

Аркушів 7

2020

АНОТАЦІЯ

Метою роботи було створення системи розвитку забруднень водяних(земних) поверхонь . Програма дає користувачу змогу вводити дані, на основі яких будується графік і користувач отримує потрібний результат . Програма має інструкцію для успішного запуску, що дає змогу використовувати її ким завгодно.

ЗМІСТ

АНОТАЦІЯ 66
ЗМІСТ 67
1. ЗАГАЛЬНІ ВІДОМОСТІ 67
1.1 Опис логічної структури 67
1.2 Вхідні та вихидні дані 67
1.3 Використані технічні засоби 67

1. ЗАГАЛЬНІ ВІДОМОСТІ

Вищеописаний програмний модуль створений у PyCharm за допомогою мови програмування Python. Призначений для отримання результату прогнозування розвитку забруднень водяних(земних) поверхонь за допомогою полыкоорди. У модулі використана бібліотека для роботи з графіками Matplolib.

1.1 Опис логічної структури

Даний програмний модуль створений для прогнозування розвитку забруднень водяних(земних) поверхонь за допомогою функціонального програмування, тобто для кожного блоку існує своя функція. Це робить код більш зручнішим для розуміння і читання. Функції зв’язані між собою, бо одна функція викликає іншу. Головна функція main при натисканні на кнопки викликає функцію press, де обробляються вхідні дані. У разі виникнення помилки вводу, вона буде опрацьована функцією validate_inputs. Якщо помилок не буде, то буде викликана функція func, яка обробляє дані, будує графіки та виводить кінцевий результат на екран.

1.2 Вхідні та вихідні дані

Вхідними даними до програмного модулю є дані, введені користувачем. Вихідними даними є графік, який будується на основі введених даних.

1.3 Використані технічні засоби

При роботі програмного модулю використовувався комп’ютер на операцій системі Windows 10 з процесором Intel Core i3 та 8Гб оперативної пам’яті. Програма була створена на мові програмування Python. Були використані пакети Math, Matplotlib, NumPy та Appjar. Розроблена програми не залежить від платформи завдяки реалізації за допомогою бібліотеки Appjar.